REPUBLIC OF TURKEY

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Information Technologies

ELIMINATE ENTANGLEMENT IN QUANTUM INFORMATION PROCESSING UTILIZING HYBRID QUANTUM-CLASSICAL NEURAL NETWORKS

Ather Salah Hadi AL-ISAWI

Master's Thesis

Supervisor

Asst. Prof. Mesut ÇEVİK

Istanbul, 2022

ELIMINATE ENTANGLEMENT IN QUANTUM INFORMATION PROCESSING UTILIZING HYBRID QUANTUM-CLASSICAL NEURAL NETWORKS

Ather Salah Hadi AL-ISAWI

Information Technologies

Master's Thesis

ALTINBAŞ UNIVERSITY

2022

The thesis titled ELIMINATE ENTANGLEMENT IN QUANTUM INFORMATION PROCESSING UTILIZING HYBRID QUANTUM-CLASSICAL NEURAL NETWORKS prepared by ATHER SALAH HADI AL-ISAWI and submitted on 09/08/2022 has been **accepted unanimously** for the degree of Master of Science Thesis in Information Technologies.

____
Asst. Prof. Mesut ÇEVİK

Supervisor

Thesis Defense Committee Members:

Asst. Prof. Mesut ÇEVİK	Faculty of Electrical and Electronics Engineering Department, Altinbaş University	___________________
Asst. Prof. Hasan ABDULKADER	Faculty of Computer Engineering Department, Altinbaş University	___________________
Asst. Prof. Atilla ÖZMEN	Faculty of Electrical-Electronics Engineering Department, Kadir Has University	___________________

I hereby declare that this thesis meets all format and submission requirements of a Master of thesis.

Submission date of the thesis to Institute of Graduate Studies: ____/____/____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Ather Salah Hadi AL-ISAWI

Signature

DEDICATION

This thesis is devoted to honoring my father's efforts, who encouraged me to work diligently for self-realization. Additionally, it is dedicated to my mother, who instilled in me the tenacity and optimism necessary to attain my ambitions. Dedicated to everyone who has aided me along the path of my modest struggle.

ACKNOWLEDGEMENTS

I'm overjoyed to be here, where I'm finishing off two years of master's studies and preparing to defend my master's thesis. I'd like to express my gratitude to Asst. Prof. Dr. MESUT ÇEVIK, my devoted supervisor, for his guidance, specific direction, suggestions, and support during the writing of this thesis. Additionally, I'd convey my appreciation to the friends and family I have.

ABSTRACT

ELIMINATE ENTANGLEMENT IN QUANTUM INFORMATION PROCESSING UTILIZING HYBRID QUANTUM-CLASSICAL NEURAL NETWORKS

AL-ISAWI, Ather

M.Sc., Information Technologies, Altınbaş University,

Supervisor: Asst. Prof. Dr. Mesut ÇEVİK

Date: 08/2022

Pages: 118

Quantum entanglement is a physical phenomenon that lies at the heart of the contrast between quantum physics and classical physics, which distinguishes quantum mechanics as an essential feature. It happens when a collection of particles interacts with one another, participates, or creates a collection of particles in close spatial proximity in a dependent way in which the quantum state of one particle in this collection cannot be characterized independently of the other particles' quantum states. The states of quantum mechanics have several types of order, including Symmetry Protected Topology (SPT), in which matter has symmetry and a finite energy gap at zero temperature. Feedback set methods are used to derive results (how to identify specific quantum data source attributes) in a more consistent manner. The concept of quantum entanglement refers to that SPT states are symmetric short-range entangled states., since only "trivial" topological orders characterize short-range entangled states.

The researcher presents and analyzes: Convolutional neural networks inspired by a quantum circuit-based technique, which is highly effective when applied within a hybrid neural network environment allowing them to be efficiently trained and executed on realistic quantum devices, this research simplifies and implements a quantum convolutional neural network (QCNN) in a hybrid neural network environment on the TensorFlow platform, where it functions as a suggested quantum equivalent to a conventional convolutional neural network.

This thesis demonstrates how to detect certain characteristics of a quantum data source, such as a device's intricate simulation or a quantum sensor. With or without an excitation, a cluster state will operate as the quantum data source, which the QCNN will learn to identify. In this study, the researcher classed the SPT phase as a dataset. The QCNN system attained a validation accuracy of 100% Instead of 89.58% in its purely quantitative analog, we use three layers of quantum convolution followed by a classical densely connected neural network to identify an exciting cluster state within 10 epochs of time. This architecture should be quite successful in mitigating entanglement.

Keywords: Quantum Entanglement, Symmetry Protected Topology, Hybrid Neural Network, TensorFlow, SPT Phase.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

<u>Pages</u>

ABBREVIATIONS

QFT	:	Quantum Field Theory
TFQ	:	TensorFlow Quantum
QC	:	Quantum Computing
QML	:	Quantum Machine Learning
ML	:	Machine Learning
QPU	:	Quantum Processing Units
NISQ	:	Noisy Intermediate Scale Quantum
API	:	Application Programmable Interface
PQC	:	Parameterized Quantum Circuits
SPT	:	Symmetry-Protected Topological
QCNN	:	Quantum Convolutional Neural Networks
HQC	:	Hybrid Quantum Classical
VQE	:	Variational Quantum Eigen Solver
QVECTOR	:	Quantum Variational Error Corrector
PQC	:	Parameterized Quantum Circuits
SVM	:	Support Vector Machines
QCNN	:	Quantum Convolutional Neural Network
PCA	:	Principal Component Analysis
VC	:	Variational Classifier

VQC	:	Variable Quantum Classifier
QAOA	:	Quantum Approximate Optimization Algorithm
CV	:	Continuous Variable
SMOTE	:	Synthetic Minority Oversampling Technique
ROC	:	Receiver Operator Curves
MLP	:	Multilayer Perceptron
ANN	:	Artificial Neural Network
SGD	:	Stochastic Gradient Descent
ReLU	:	Rectified Linear Unit
DMRG	:	Density Matrix Renormalization Group
PEPS	:	Projected Entangled-Pair States
MPS	:	Matrix Product States
TTN	:	Tree Tensor Networks
CAS	:	Computer Algebra System
HQCNN	:	Hybrid Quantum Convolutional Neural Network
CPU	:	Central Processing Unit
QPU	:	Quantum Processing Unit
GPU	:	Graphical Processing Unit
TPU	:	Tensor Processing Unit
RBM	:	Restricted Boltzmann Machines
EPR	:	enhanced permeability and retention effect

MERA : Multi-Scale Entanglement Renormalization Ansatz

LIST OF SYMBOLS

$U_{\Phi(\vec{x})}$	:	Unitary Gate
w	:	Isometries in Every second row of TENSORS
$\overline{L}$	:	Logical Pauli Operators
$S(\rho^{[B_l]}$	:	Entanglement Entropy
$\left\langle \Psi_{\{\phi_r\}} \middle\| \Psi_{\{\acute{\phi}_r\}} \right\rangle$	:	Encode Quasi-Particle Excitations in only one (Generalized) MERA
$\|\psi\rangle$	:	Represent The State
α	:	Angle in Bloch sphere
$\boldsymbol{v_\phi}(\overrightarrow{\boldsymbol{x}})$	:	Feature Map
$\boldsymbol{H}$	:	Hadamard Gate
$\mathcal{H}_2$	:	Hilbert Space
β	:	Angle in Bloch sphere
$U^\dagger$	:	Unitary Gate with conjugated transpose
$\mathcal{R}_x(\alpha)$	:	the Bloch sphere with rotation operators as single qubit unitary gates around x axis with the angle $\boldsymbol{\alpha}$
σ_x	:	Matric of Pauli's shorthand notations to *x*
G	:	$excitation\ graph$
ρ_j	:	the current state
$\mathcal{E}_{n,0}$	:	deducing trace-decreasing operations

$\boldsymbol{\theta}$	:	discrete time
$\boldsymbol{\mathcal{L}}$	:	lattice
$\lvert\boldsymbol{\Psi}\rangle$	:	the N-site state
$\mathcal{M}$	:	(*MERA*) Entanglement Renormalization Transformations when reversing the arrow of time $\boldsymbol{\theta}$
CSS	:	Color Code Stabilizer Group
S_{TC}	:	full Code Stabilizer Group
S_{CC}	:	Sub Code Stabilizer Group
H_{CC}	:	The color code' stabilizer Hamiltonian with a twist
H_{TC}	:	The toric code' stabilizer Hamiltonian with a twist
e_D	:	The Excitation of the same kind using geometrically local operators or destroyed in pairs
v	:	VERTICES

1. INTRODUCTION

The particle physically (as called in the ancient text the corpuscle) is an object that can be associated with several physical or chemical properties, such as mass, density, or volume, unlike the electronic particles, and microscopic particles for instance atoms and molecules, and powders and other granular materials, in vary greatly in quantity or size [1]. According to density, these particles can be used to create scientific models of even larger objects, such as celestial bodies in motion or humans moving in a crowd [2]–[4]. In theoretical physics, there is a paradigm that says that quantum field theory (QFT) is critical in a theoretical framework that incorporates quantum physics, classical field theory, and special relativity. Undoubtedly, in condensed matter physics and particle physics, QFT is employed, and therefore the consequences of this led to constructing physical models of subatomic particles, to constructing models of quasiparticles [5]. In classical field theory, Lagrangian field theory is a formalism, it is Lagrangian mechanics' field-theoretic counterpart [6]. The motion of a system of discrete particles, each having a finite number of degrees of freedom, is studied using Lagrangian mechanics [6]. The Lagrangian field theory is applicable to fields and continua with an unlimited number of degrees of freedom [6].

Feynman diagrams are a visual depiction of the mathematical formulas that characterize the subatomic particles' behavior and interactions. These diagrams were presented by the American scientist Richard Feynman in 1948, these diagrams gave a simple visualization of abstract, mysterious equations, these graphs can be used in many fields, including the quantum field, the solid-state field, and others [7]. In theoretical particle physics, the computation of the probability amplitudes requires the use of complex integrals to a large extent on a vast number of variables, and these integrals can be represented graphically by these diagrams [8]. The above leads us to the fact that quantum field theory can be represented graphically using Feynman diagrams in order to clarify the correlation function of statistical field theory, transmission amplitude, or quantum mechanics [7]. This leads us to the conviction that interaction terms involving corresponding quantum fields as described by Lagrangian as playing the role of interactions between particles [6], and within the quantum mechanics' perturbation theory, it is possible to mimic graphically each of these interactions. by means of Feynman diagrams, where particles are considered as stimulating quantum field states which they belong [9].

Quantum computing solves issues by calculating utilizing physical events and quantum mechanics concepts. Theoretically, this type of computing has been proven to give speedups to various modern-day processing challenges. Quantum phenomena' applicability in the field of Machine Learning has been extensively expected. The work presented here generates machine learning models using the TensorFlow Quantum (TFQ) software platform. In a quantum learning model, developed models use an information encoding technique called amplitude encoding to prepare states. The introduction of these novel approaches raises the question of how to effectively employ them. The researcher hopes to provide an overview of the current status of quantum machine learning in light of actual device restrictions. Although quantum information does not modify the procedures of data preparation, it does require new ways to interpret and appreciate these innovative technologies.

Quantum computing (QC) combined with machine learning (ML) is one of the major areas of interest in Quantum Machine Learning (QML) field. To provide accelerations a number of problems in the field of quality control (QC), many algorithms are available such as Grover's and Bernstein-Vazirani, among these problems are integer factor analysis and database search by Quantum Processing Units (QPU). The NISQ instruments were extended to include finance, chemistry, biology, among others. NISQ devices are testing and developing QML technologies while developing many programming groups in the NISQ era of quantum simulators, which allow a richer experience and try to understand realistic expectations of systems but on a real quantum device. In order to achieve Application Programmable Interfaces (APIs), these programming groups will be divided into two main parts:

a. Section one: the supporting code for circuit development that enables models to be developed on both simulated quantum devices and real quantum devices, as it has been adopted by major organizations like Google and IBM. It is possible to construct quantum algorithms with basic applications of the device's neutral gate using the developmental code.
b. Section Two: the compilation code which implements on the physical device.

The python programming language which is supported by IBM and Google can be used to implement a set of gates in a quantum circuit with the aim of making a modification to a quantum state's computational base(s).

There's also the "backend" which has to deal with both software and physical issues. During the execution of an experiment, the backend manages elements such as noise and, qubit mapping to device topology, and transcoding. These backend stages are handled differently by each of these code sets. IBM's Qiskit & Q Experience, Xandu's PennyLane, and Google's TensorFlow Quantum are some of QML’s prominent participants who have produced open-source frameworks. Each of these frameworks may use tertiary APIs to extend their software suites and use their frameworks on devices that aren't owned by their individual corporations. There is a corpus of work that will be required to respond to the following questions:

How can we increase our program's performance?
When should we utilize an embedding approach rather than an encoding method?
What caused a model's performance to be as it was?
One of the best methods is amplitude coding method, which is used to encode raw classical data into its quantum counterpart. In addition to confronting the problem of quantum entanglement occurring and trying to reduce it, which is the subject of this thesis research.

1.1 PROBLEM STATEMENT

From a fundamental physics point of view, simulation concepts have been deeply developed by researchers in recent times. Because of the recent interest in quantum devices and their efforts to answer the following question: Can a physical system efficiently simulate another system? This is because these devices have superior features to simulate any other physical system with excessive efficiency. Turning to the components of these quantum devices in terms of infrastructure related to practical programming, the following question is raised: Can the infrastructure components of quantum devices meet the aspirations of their users? The answer to this question requires delving into the statement of the basic units of the components of quantum devices, and how this structure is used programmatically in solving quantum problems. Machine learning techniques offer an ideal solution for analyzing complex quantum multibody systems. The working principle of many variable quantum algorithms is based on parameterized quantum circuits (PQCs), where the selection of an effective circuit is one of the most important challenges in implementing such algorithms, considering the most important conditions that must be met in reducing the depth of a low circuit as well as reducing the number of parameters in order to optimize the results [10], [11].

Despite the success of neural networks in the application of classical algorithms, as well as in image processing operations, which are considered classic problems, leading to the improvement of classical error correction, but the enormous complexity of the systems of many objects prevented the ease of direct application to problems in quantum physics, which forces work in parallel with the principles of quantum physics concepts [10], [12]. This led to the use of quantum states or operations as inputs to classical algorithms in order to reduce the depth of the circuit and the number of parameters due to the inability to effectively translate classical problems into a quantum framework or vice versa. The reason lies in the extremely large and multi-body Hilbert space [10].In addition to the imperative to benefit from the accuracy of the results expected to be obtained using quantum algorithms, as well as to take advantage of the speed of completing tasks as one of the advantages of classical algorithms [10]–[14].This leads to facing the problems of limited physical resources in quantum devices and the large quantum entanglement that occurs in these devices [15], [16].

1.2 MOTIVATIONS

By leveraging the programming power of Python and using TensorFlow Quantum (TFQ) as a framework, the advantages of this programming language do well in the field of ML and leverage Google's quantum computing frameworks, then hybrid quantum- classical models can be built by focusing on a Quantum data [17]. TensorFlow provides tools to interleave quantum algorithms and logic which in turn help build Cirq [18].Since a quantum data source suffers from quantum entanglement, as in a complex simulation from a device or a quantum sensor, a cluster state can be considered as a quantum data source (Symmetry-protected topological (SPT) order phase classification) that may or may not contain an excitation that is detected by QCNN algorithm [1], [19]. In addition, the quantum convolutional neural networks (QCNN) are derived from machine learning focusing on the fidelity of prepare a “better-learned” trial state as results [12], and then fed the results into a classical neural network as followed by a densely-connected neural network [10]. This leads to faster convergence in the hybrid model with very modest classical assistance than its purely quantum version. While using an architecture with multiple quantum convolutions within the hybrid network will give higher validity accuracy with performance. The rapid and exponential development of both hardware and algorithms brings us closer to devices’ era called noisy intermediate scale quantum (NISQ), this generation enhances approximately around 103 gate operations within 50−100 qubits [20]. Certainly, these configurations are unable to execute error-corrected quantum calculations on a large scale, to employ modest amounts of computing power, both quantum and classical resources are merged, meaningful computations. This leads researchers to divide large tasks between quantum and classical resources by using the hybrid quantum classical (HQC) algorithm, where in turn focuses the use of devices on such pre-threshold hardware. An HQC algorithm is exemplified by as a prime example to account validation accuracy for a cluster state.

HQC algorithms are used in a variety of ways.

i. the approximate quantum optimization algorithm [10], [21].

ii. the variational quantum eigen solver (VQE).

iii. quantum autoencoder (QAE) [10], [22].

iv. classification via near-term quantum neural networks [10], [23].

v. quantum variational error corrector (QVECTOR) [10], [24].

vi. energetic modeling.

vii. used to calculate molecular systems' ground states.

A quantum circuit is adjusted based on the parameters chosen on the quantum computer to set up a parameterized wave function. This is followed by measurements of the validation accuracy for a cluster state as expectation value with respect $\langle \hat{Z} \rangle$ to benchmark the model.

1.3 OBJECTIVES AND GOALS

This research aims to reduce the quantum entanglement that occurs in the cluster state of quantum information by increasing the accuracy of the results and increasing the speed of performance in discovering certain properties of a particular data source such as a complex simulation of a quantum sensor or device, by building efficient and optimal quantum circuits with coefficients (parameterized quantum circuits PQCs) for block-state while keeping circuit depth and parameter count minimal based on quantum circuits are inspired by highly entanglement convolutional neural networks and implemented in a kind of neural network that utilizes variable parameters O (log

(N)) only for input sizes of N qubits, making it easy to train and deploy, that called a the quantum convolutional neural network (QCNN), after that train the layers on the fidelity of preparing a “better-learned” trial state as a results, and feeding the results into a classical neural network as followed by a densely-connected neural network. Due to QCNN’s ability to accurately identify quantum states linked with a symmetry-protected one-dimensional topological phase (SPT), and using the CNN property known as fast performance, it is possible to achieve faster convergence in the hybrid model and give higher validity accuracy with performance directly proportional to the number of filters provided by HQC. This gives a quantum error correction scheme that is significantly superior to other known quantum codes of similar complexity as a result of optimizing a particular unknown error model. By leveraging the programming power of Python and using TensorFlow Quantum (TFQ) as a framework, the advantages of this programming language do well in the field of ML and leverage Google's quantum computing frameworks, then classical hybrid quantum models can be built by focusing on a Quantum data. Thus, a number of objectives will be achieved in this research including:

Conduct a comprehensive literature review of current studies on quantum physical information processing and deep learning methods used in solving classical and quantum problems and combine them within a single work environment:

i. Utilizing a cluster state as a source of data that may or may not be stimulated and interpreting quantum entanglement in Symmetry Protected Topology (SPT).

ii. Suggesting a quantum analogue to a classical convolutional neural network represented by a simplified Quantum Convolutional Neural Network (QCNN).

iii. Discovering certain properties of a quantum data source, such as a complex simulation of a quantum sensor or device.

iv. Activate the role of the TensorFlow work platform, which has a comprehensive ecosystem of tools that allows researchers to solve difficult real-world problems using machine learning and take advantage of Google's quantum environment.

v. Develop a method for reducing quantum entanglement by giving high accuracy in validation accuracy within a specified period of time (epochs) at a high speed.

vi. Conducting a comparison between the proposed methods and the related studies to evaluate the results.

1.4 THESIS STRUCTURE

i. Chapter one illustrates introduction about the particle physically in theoretical particle physics and the interpretation of related phenomena within quantum field theory (QFT), In the Theoretical Background portion, build a rudimentary introduction to conventional data processing methods, machine learning methods, and quantum computing. In the Technical Background section, describe the technologies utilized and the individual frameworks for quantum machine learning and, problem statement, motivations and aims and objectives of the research, also it shows how to reduce quantum entanglement by leveraging the programming power of Python and using TensorFlow Quantum (TFQ) as a framework.

ii. Chapter two discussed some relevant literature and traced the development path of the thesis research field in the historical perspective of quantum computing and focused on the most important problems that researchers could face during the development of the research field.

iii. Chapter three was interested in explaining the primary concepts and basics related to the field of thesis research by clarifying the background of quantum theory and how to process classical data and transform it into quantum counterparts, focusing on all topics, including normalization, analyzing results using machine learning, and how to teach the quantum model in order to reduce losses by making a link between classical learning and quantum learning on models. The concepts of parameters and quantum gates applied to cluster state models in the simulation process for preparing the state and explained the role that MERA plays in DEEP QCNN ARCHITECTURE down to the measurement of Quasiparticle excitations.

iv. Chapter four was interested in explaining how to use the work environment on the workspace TensorFlow Quantum that supports the Python programming language, and the researcher deliberately clarified the ready-made functions and libraries in this programming language that TensorFlow Quantum supports after initializing the Python version compatible with algorithms hybridization supporting his work by citing some pieces of the applied code in support of that A set of Figures that show the sequence of events during the execution of the code and giving the results.

v. Chapter five Focused on a Quantum data Flowchart with a hybrid quantum-classical discriminative model's training and inference using TFQ's pipeline and discuss the results.

vi. Chapter six revolved around the summary, future visions and recommendations of the researcher.

vii. Chapter seven contains the references.

2. THE HISTORICAL PERSPECTIVE OF QUANTUM COMPUTING AND RELATED WORKS

Literature on the history of quantum computing has a variety of hypotheses, making it difficult and time-consuming to arrive at the current discoveries due to divergent perspectives on the principles of this remarkable topic. Much of the present literature on quantum computing focuses on the possibility of a qualitative leap in humankind's future and the modification of many previously held scientific beliefs. In past, the generalizability of much-published research on this issue is problematic, where the academic literature on quantum computing has revealed the emergence of several contrasting themes, where previous research findings into Quantum Circuit Simulation have been inconsistent and contradictory also a large, and growing body, of Literature, has researched how to take use of the features of particular superconducting components to represent these quantum circuits, and there is a substantial number of published publications outlining the function that these elements play in the system. More recent attention has focused on the provision of Efficient algorithms that play an important role in creating these quantum circuits. As well as combining classical and quantum representations in a single setting, there is a large body of literature on these algorithms, most of which emphasizes using the unique characteristics of superconducting materials. A considerable amount of literature has been published on these algorithms. These studies were as illustrate bellow:

The study of quantum computers has a lengthy history of investigation. Prior to Feynman’s work (1948) [7], the role of exploring quantum computing's theoretical was largely unknown. For many years, this phenomenon was surprisingly neglected by his peers where Feynman presented his results and the foreign graphics at the Pocono Conference in 1948 under the title "Alternative Formulation of Quantum Electrodynamics" [25]. In the 1980s, Paul Benioff's paper provided the first in-depth description and analysis of a quantum mechanical model of Turing Machines [26], who he was exploring quantum computing's theoretical feasibility in the 1970s. It is only since the work of Yang-Mills theory [27] that the study of quantum information and mirror symmetry has gained momentum where Yuri Manin has also written about them [28], after many years, Deutsch and Jozsa were the first to describe the first algorithmic structures as a consequence of their contributions [29]. Peter Shor (1994) contends that the interest of applied mathematicians in integer factorization and discrete logarithms was sparked which caused the advent of the so-called

Short's Factoring algorithm [30], Collectively, these studies outline a critical role for speeding up cryptanalysis methods, has put the safety of communications and data storage in jeopardy. Moreover, during the actual implementation of quantum computers, developing technologies confront additional obstacles, such as the rapid loss of information to the environment, together these studies provide important insights into the quantum gates, due to a phenomenon known as decoherence [31]. However, such studies remain narrow in focus dealing only with the work of certain algorithms is limited. These research together provide light on how the use of superconducting metals permits the fabrication of resonant circuits with millisecond-long coherence durations, bringing quantum computers closer to reality, yet noise remains an intrinsic component of quantum computers [32]. Simulations and algorithms appropriate for today's hardware and technology are built using specified algorithms that can be executed with existing technology on Noisy Intermediate Scale Quantum Devices (NISQ) [20]. In comparison to their classical counterparts, quantum computers are capable of monitoring and characterizing qubits, but quantum understanding would need a huge collection of classical numbers [11]. Quantum mechanics' postulates are algebraic in nature, indicating that algebraic operations and quantum computing are inextricably linked.

As a result of the creation of more varied and significant applications around the early 1990s, small-scale research and case studies by Engineers and scientists began to develop the quantum circuits, and quantum twin of the binary bit are linked by the conception of qubits. Given that NISQ devices lack practical realism at that time [20]. Up to now, most research of several studies on NISQ devices has been carried out in implementing various data processing and machine learning methods using this approach, demonstrated by researchers, engineers, and scientists that able to increase the exponential speed-up techniques in Principal Component Analysis (PCA), Support Vector Machines (SVM), and K-means Clustering, resulting in multiple advances in current Quantum Computing [33].

In 2019, Rebentrost et al. invented the quantum counterpart of gradient descent as two aims: To reduce the size of the function as an iterative improvement method based on Newton's gradient descent method, taking into account the curvature information, where the number of iterations for high-dimensional cases was reduced in order to improve convergence [33]. The authors look at a class of polynomials that are restricted due to sparsity limitations, which means that the optimizations may be applied to functions of lower order. The authors also state that the vectorized data's input dimensions should be in a binary space, or 2^N. They used to analyze curvature according to Newton's approach to avoid orthogonal movement in analyze curvature. This approach makes use of the descent being projected onto spherical restrictions. Considering independent variants of their quantum stochastic gradient, they build three quantum oracles [34]. Under spherical restrictions, Hamiltonian simulation methods have improved a class of polynomials bound by extending their approach under these restrictions. They claim that since the optimization approach uses Newton's method, whereby using 5-25 iterations it is possible to obtain a solution to any convex issue with high accuracy. As well as they bring up challenges that are comparable to those encountered such as "saddle spots" in classical optimization where high-dimensional space in Newton's technique. They conclude by claiming that, theoretically, their technology would enhance conventional gradient descent-based methods exponentially [34].

In 2019, a seminal article was published entitled "Supervised learning with quantum-enhanced feature spaces" by Havlek et al. who proved that two binary classifiers that use feature spaces for the quantum state may read normally provided data [35], [36], they suggested that as:

i. The first technique uses a variational quantum circuit with a binary measurement.

ii. To estimate a kernel method, the second strategy is based on the conventional SVM, which uses hyperplane construction.

In order to achieve great accuracy, they adopted a non-linear kernel approach when they used a 5-qubit quantum processor from IBM. They achieved 100% accuracy on the created dataset [14]. This is a good example of machine learning in the NISQ era. Where four phases were defined for a variational classifier (VC):

a. Use a feature map to map data to a quantum state.
b. To the feature state, apply a brief quantum circuit.
c. Apply a Z-basis measurement or a *Z*-gate measurement.
d. A decision procedure has been applied to generate an empirical distribution of outcomes by doing numerous "shots" or runs and assigning the label with the greatest probability [35].

Due to the noise and current limitations of NISQ era devices, their solution relies on the concept of shots. This evidence, it seems too close to our work.

In 2020, Schuld and Lloyd offered a quantum embedding strategy for improving learning performance in high-dimensional Hilbert spaces by a paradigm change in how we think about optimizing a model, Where, instead of fitting classes to a local function, it maximizes two separate classes as quantum modulation [36]:

i. The classical samples are converted into quantum states using the map of quantum properties generated by the first component [36].

ii. The second component reduces the classification loss after knowing the required objective function as it causes the model to re-quantify [36].

A parametric circuit optimizes the quantum feature map, a parametric circuit optimizes the quantum feature map, The Helstrm scaling is described, and the accuracy scaling is described, leading to the data being characterized as a Variable Quantum Classifier (VQC) [37]. To ensure that the accuracy or experimental risk is kept to a minimum, the loss function or the Hilbert-Schmidt distance is improved while performing state inversion of samples through the SWAP gates that make up the fidelity measure [36].

Previous studies have explored the relationships between the classical model and the quantum model. Researchers attempted to evaluate the impact of combining in a quantum-classical model. What the authors show in classical quantum models is the apparent integration of their methods depend on the model of Quantum Approximate Optimization Algorithm (QAOA) to controlling set of model weights via deep learning represented by utilizing ResNet [36], and those methods supported our work.

In 2019, Killoran et al. introduce a fully connected layer principle based on parametric quantum circuits to build quantum neural network topologies with Continuous-Variable (CV) [38]. They show that a CV design is useful for creating quantum-classical networks by connecting conventional neural networks with a quantum neural network in a single system. CV design can also be used in other types of networks, such as residual networks (ResNet) and convolutional networks. Where they used four training models, two of these models were applied hybrid quantum [38]. This also supports our work on this thesis.

The classic computer has practically proven its effectiveness in solving a huge number of tasks with infinite accuracy in the results, but despite the classic computer's good performance, the problem of speed in obtaining results remains a factor that many researchers and computer designers are interested in. This is what quantum physics promises, which tries hard to give a scenario that can be summarized by exploiting the physical properties of some materials used in the manufacture of computer devices and by introducing designs for the circuits that make up these devices that are characterized by the use of modern quantum techniques. The application focuses on converting a classical state to a quantum one by constructing or programming quantum circuits. When transforming a small number of classical bits into comparable qubits, a huge amount of space is required. In practice, the importance of a classical computer in accomplishing tasks is not erased, since the ability of classical algorithms plays a part in dealing with a quantum computer. Both conventional and quantum devices' work may be regarded as collaborative and complementary. Quantum devices handle complicated mathematical operations that would be almost difficult for conventional devices to handle. Quantum devices, for instance, can handle Hilbert space. A Hilbert space is a complete space that is a vector space extension with an inner product structure. In the term of the inner product's distance function, a Hilbert space is a complete metric space. It might be simple or complicated. In contrast to quantum devices, which use linear algebra in quantum gates comparable to matrices, classical devices may consider individual values as a unit of arithmetic using Boolean algebra that acts on logic gates. When it comes to quantum processing, one of the requirements is to conduct base change generalizations and linear operations, which Hilbert space provides.

The subject of quantum device development is still in its early stages, and NISQ devices are now being used to prepare, test, and create quantum algorithms. Stochastic noise, which renders quantum devices prone to inaccuracy, is one of the issues confronting this generation of quantum devices. Stochastic noise is a non-ideal circumstance that necessitates study in the area of quantum error correction to improve the accuracy of quantum systems. It's worth mentioning that when using the lithographic method to make quantum devices, a Josephson junction is generated between the qubits, which is considered a physical link. A tunnel produced by two superconducting metals separated by an insulating barrier forms this junction. Quantum tunneling is the name for this kind of tunnel. The researcher emphasizes that IBM quantum devices have a distinct number of qubits and topological connectivity, which is not directly connected to their processing capabilities but rather to the term quantum volume. CNOT error, U gate error, crosstalk, topological connection, circuit depth before the degree of error, and the number of qubits all are variables that influence quantum volume.

3. THE BASICS OF INTERPRETING THE INITIAL CONCEPTS

To address issues, quantum computing employs physical events and quantum mechanics concepts. Theoretically, this kind of computing may help solve some of today's processing challenges. It has become clear in the field of machine learning the use of quantum phenomena, which is something that many people are happy about. The work presented here creates a model based on a software framework to represent machine learning using TensorFlow Quantum (TFQ) by building optimum quantum circuits in converting classical data into quantum data, taking advantage of the accuracy of quantum algorithms (QCNN), and making twists of their outputs to feed the classical algorithm (CNN) with the aim of obtaining the required accuracy in evaluating the results after reducing the excitation occurring in particles and thus reducing or lacking entanglement. This thesis investigates in a quantum learning model the ability to encode information using an amplitude coding approach using a Variable Quantum Classifier (VQC) as one of the learning techniques to give an improved preparation of the state by providing insight into the training data by characterization during deep data examination [39]. The introduction of these novel approaches raises the issue of how to effectively use them. The researcher intends to provide an overview into quantum machine learning's present state in light about real-world device constraints. The results of utilizing hybrid quantum-classical neural network over TFQ indicate that amplitude encoding has a clear advantage over other approaches. In addition, the no-free lunch theorem in a VQC is similar to conventional approaches that increase the number of learning rounds leading to feature- rich data [39].

3.1 THEORETICAL BACKGROUND

This extremely specialized sector of quantum machine learning QML is made up of various components. This section's goal is to providing readers with sufficient background information by lightly discussing many of these issues before moving on to a more detailed look at the issues.

3.1.1 A Classical Data Processing Overview

Data preprocessing and data analysis are critical for establishing a thorough understanding of the material that will be examined or learned from. The researcher will go through a few of these approaches that may be employed before and after using a learning strategy, as well as methods for understanding the outcomes.

3.1.2 Dealing With Raw Data In Its Entirety

A system's or program's raw data is frequently gathered for no other reason than its intended purpose. Data can be in a variety of formats, containing non-structured text, categories discrete variables, binary variables, continuous real variables, and multimedia (audio, pictures, and video). Data is frequently not ready to be learned from or used analytically in its raw form. It can be difficult to even get a basic understanding of the data without simplifying and streamlining the data using certain procedures to the desired level. Missing records, unbalanced classes, and outliers are three difficulties that are frequently connected with data processing. Missing records are a challenging challenge to solve since they would otherwise provide useful information for a learning process. The missing data problem requires the development of a number of strategies, for example using the feature's mean instead of the data itself, however, this results in the loss of potentially relevant material. What is worrying when dealing with data is unbalanced classes. This becomes clear when using heuristic learning, when one class or group of classes contains more samples than others, then the classes are unbalanced. One class's population outnumbers the others. In the worst-case situation, a technique may just evaluate all facts in favor of the majority. To solve this problem, using *Oversampling* methods by eliminating entries from the majority class and replacing them with records from the minority class, data may be balanced, such as the Tomek Links and Synthetic Minority Oversampling Technique (SMOTE) [39].

Outliers' third issue is generally more difficult to analyze and deal with. When data is found on the outskirts of the sample distribution, it is referred to be an outlier. Simply expressed, data that does not follow the same general pattern as the rest of the data. This may be managed in the simplest instance by removing samples with more than a certain number of standard deviations.

3.1.3 Feature Dependent Processing

Direct data transformation and manipulation procedures are frequently required in the preprocessing step to clarify the information in specific characteristics. Discretization, normalizing, and smoothing are a few of these techniques. When using QML approaches, A model's learning behavior is replete with many modifications as a result of these techniques. [39].

3.1.3.1 Discretization

In general, raw, continuous variables must be transformed in a feature discretization way into their counterpart across a separate, less complex feature space, which in itself is a devaluation. Here are two interesting questions: When should a discrete set or interval be stopped or started? How is an interval or discrete set determined?

The data is divided into m-bins of size after sorting in a method called discretization, after which this data is translated based on the mean of each bin into bin values [39], This approach has trouble determining what size *m* should be in order to get the greatest results, and it may take several cycles of trial and error. The second method is three steps to discretization based on ChiMerge Technique, first of all, the initial interval is set after the data is sorted ascending (each pair of values is considered) where for each time interval there is only one value. The second step is calculating a value about X^2 for each adjacent interval and compare it with the threshold to see if X^2 is less. The last step is to merge the two periods in case X^2 is less than the cut-off otherwise the intervals are not combined. The boundaries of a newly merged interval will be replaced by the bottom and higher bounds of the first and second intervals, respectively.

To utilize ChiMerge technique, to compute the values utilized in the X^2 computation, it is necessary to establish a contingency table that employs the classes' number in the dataset. The ChiMerge technique is considered an statistical strategy, the benefit of this technique is to perform classification due to the large difference that it presents in the different results when choosing the intervals using the binning method [39].

3.1.3.2 Normalization

Data is normalized when it is scaled between a specified range, such as $[0, 1]$. The values are ordered using a technique that considers all of the samples for a single characteristic. As a result, if applicable, normalization occurs column-by-column over the whole dataset. This is due to the fact that the values of each characteristic are unrelated to the values of the others. The idea of normalization is to reduce the influence that a single value might have on the behavior of a learning approach when viewing a sample. Misguided learning under certain circumstances produces unusual (large/small) values, which (increases/decreases) the importance of feature values for these data, and this leads to limiting normalization to the data set with a majority of small data sets, so the outliers of the data must be deleted before normalization. This method has disadvantages, given that the learning technique may encounter difficulties in interpreting the characteristics of other data sets in the presence of outliers during the process of narrowing the data range during normalization [39]. They are weighed in on the scale for normalization if they are included. Normalization without eliminating outliers might produce inaccuracies that become more difficult to understand as the study progresses. Normalization can take several forms, including standard deviation normalization, minimum-maximum (min-max) normalization, and decimal normalization. Table 3.1 displays these equations.

Table 3.1: Shows data scaling normalization methods [39].

Normalization Methods	Min-Max	Standard Deviation	Decimal
Equations	$\frac{v_i - \min(v_i)}{\max(v_i) - \min(v_i)}$	$v_i - mean(v)$	$\frac{v_i}{10^k}$

Before and after the learning process, the data must be evaluated since the normalization and application process depends on this data. To better match the data, a number of strategies are modified, such as the standard deviation and decimal normalization, as is evident in the above table, it is obvious that v_i represents the value of the column when the column being normalized is v. The standard deviation of the std is calculated by finding the value of k, which is the amount of force required to make the column's greatest value equal to or less than one, in the decimal settlement, to fit the data appropriately. Adjustments are made to these strategies. More precisely, the weighting of the values may be reduced, and a parameter applied to the denominator in order to use the standard deviation normalization or depending on the classification task by doing a minimum-maximum normalization by normalizing between $[0, 1]$ or $[-1, 1]$, before and after the learning process, when the data must be evaluated, as it is normalized and applied to this data in both cases.

3.1.4 Analysis Of The Results

In order to explain results from previously applied techniques, ex-post analysis after training the model is an important component, the best way in almost every case (the classical or quantum field) is to analyze the waiting method of learning or to use and test samples that have not yet been used anywhere in the learning process. There are two modes of post-analysis:

a. The data sets are separated into categories: training and testing.

b. The data sets are separated into categories: training, validation, testing.

As it will be clear, During the learning process, feedback is provided based on the accuracy of the intermediate results, which is called a validation set accessible by the technology and which is not recommended for testing. The model must never use the holdout or testing set again where the method of processing the holdout set is similar to the method of processing the data [39]. As a result, the same preparation approaches must be used, such as normalizing and discretization. In terms of actual application, it's better to separate the testing set from the data just before starting any training, since this ensures that all of the essential processes have been appropriately followed. The learning model's accuracy on the new testing set is the most typical statistic employed here. When assessing a model's performance, the researcher examines $F1$-Score, Accuracy, Recall, Receiver Operator Curves (ROC), and Confusion matrices. also, the researcher explains how these measures suggest that employing TFQ to prepare an amplitude encoded dataset beats alternative state preparation approaches significantly greater.

TFQ is superior to its peers in case preparation methods, and this is illustrated by the learning outcome of an encoded data set by focusing on capacity metrics. In addition, it was found that the metric results that were collected change dramatically in the quantum models when making different transformations of the data. Despite this, the evaluation will remain in the thesis using all scales, as accuracy was an appropriate choice depending on the behavior of the data set that was handled. Given that Post analysis of data interprets the outcomes of the learning technology and not the techniques used in the learning process, it can be considered deceptive. While post-analysis is instrumental in demonstrating forms of encoding and transformations, which in turn teach data in a quantitative space/representation. Where it is possible to capture the behavior of technology in the graphical results in some cases which makes these numerical metrics of great use. So, Data post-analysis might be challenging since it tries to explain the learning technique's output.

3.2 BRIEF OVERVIEW ON MACHINE LEARNING

Artificial intelligence is paving the way for deep learning with new devices based on the complexity of learning in the modern era. Which led to the survival of many of the basics of learning unchanged, the researcher tries to make a brief overview to clarify some of the basic components of machine learning, as well as discussing some simple methods that still play a very effective role. In a broad sense, modern learning methods have become extremely sophisticated, the researcher has gone through several simple *strategies* that, despite their age, still work effectively. *Prediction, classification, and clustering* are the three categories of machine-learning algorithms. These three kinds, while complicated in their varied implementations, may be explained in a straightforward manner as follows:

a. *prediction* is a job that seeks to determine a value given a collection of inputs with some degree of precision or accuracy. The accuracy of a prediction is tested against the prediction's immediate outcome.

b. *Classification* is the process of determining which class a sample belongs to. The number of successfully categorized samples determines the accuracy of a classification model.

c. *Clustering* techniques are typically based on a distance measure that looks at a "spatial" aspect of the data, i.e., those who are nearer to one another in n-dimensional space those are closer or geographically nearer to each other are clustered together. Clustering is a descriptive strategy that is commonly used before prediction or classification to generate some quantifiable by ordering or grouping subsets together within a dataset, an explanation can be found.

The *techniques supervised and unsupervised learning* creates further distinction in machine learning concepts.

i. *Supervised learning* is a way of feeding data samples to a procedure that has a class label or predicted outcome. The purpose of any supervised learning approach is to let the process run, apply whatever methods are available to it, and make an educated prediction about the predicted result based on its existing level of knowledge. The outcome (or group of results) of these estimates is then compared to the output or compared to the genuine internal functions and the model parameters are often fine-tuned to the process starts all over repeatedly [39]. that essentially to illustrate the best machine learning and deep learning approaches get their best outcomes.

ii. *Unsupervised approaches* are often used to provide formal descriptions of data, as the problem is approached differently as it occurs in such as clustering approaches, since their applications are different from what happens in supervised learning. It is not possible to measure part of the learning or after iteration using Unsupervised learning. It is not possible to be certain that clustering approaches are the same as Unsupervised learning. When clustering techniques are applied, they are generally the best performing among many techniques. Unsupervised approaches are often used to provide formal descriptions of data. The apriorist algorithm is an example of market-basket analysis while the Restricted Boltzmann Machines (RBM) is an example of learning probability distributions of a dataset.

iii. *Semi supervised learning* is a mixture of unsupervised learning and supervised learning.

3.2.1 Methods Of Learning

The perceptron, also known as a *multilayer perceptron* (MLP), has cleared the way for deep learning. In 1958, the perceptron was invented. The researcher used an MLP to make a quantum-classical method. Perception is a basic function that takes a vector with its' dot product that contains a real-valued weight as inputs [39]. A perceptron is extremely basic, and as a result, it cannot tackle nonlinear issues. Taking a simple example illustrated in, we conclude that a simple perceptron will not be able to separate orange points from red triangles, in other words, XOR gate faces a classic problem that cannot be solved by hyperplane. Which calls for devoting several concepts together to form a “layer”, and that An MLP is made up of numerous layers (usually three or more). An MLP is a dense layer or fully connected layer since each of its nodes (neurons) is linked to all others, so it used to learn Nonlinear approximations [39] as shown in Figure 3.1.

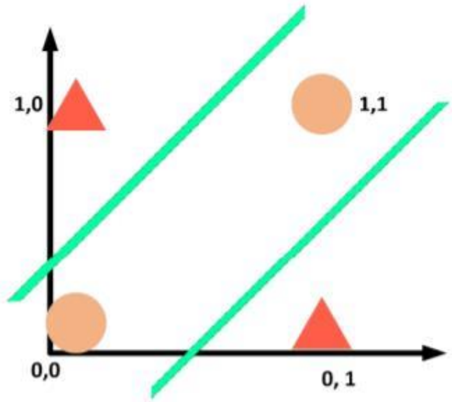


Figure 3.1: A classic XOR gate example A classic XOR gate example demonstrating the inability of a basic perceptron to address nonlinear issues. MLP fixes the issue via grouping samples within and outside the boundary (dots and triangles) respectively [39].

One of the fundamentals of deep learning is working in several overlapping "layers" of MLPs that may number into the thousands. Where it is divided in terms of processing methods into three parts: input layers, hidden layers, output layers. Figure 3.2 shows an example wherein a network of neurons consisting of 16 input neurons, hidden neurons, and one output neuron.

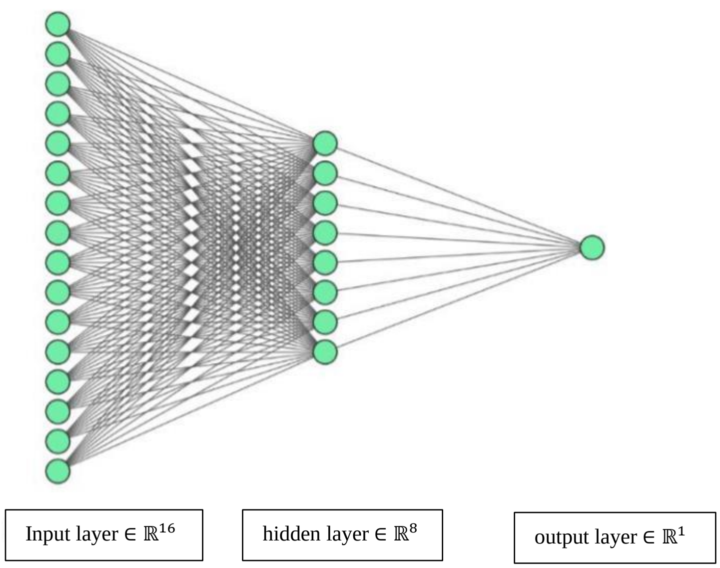


Figure 3.2: Input, hidden, and output layers of an Artificial Neural Network (ANN) having perceptrons with numbers 16, 8, 1 respectively. In TFQ experiments, a network with nodes number two times in the second layer were employed [39].

Where software packages enable us to make modifications to the output layer of a perceptron, the activation function is used to control binary classification or continuous value regression as well as multiclass classification.

3.2.2 Learning – Optimization And Loss

A basic factor in all learning systems is optimization, which can range from modest to extreme. The notion of learning is breathed in the adopted strategies via optimization. In fact, optimization entails finding a linear or non-linear equation in a space of feature, for instance, $y = ax + b$ in the linear case [39]. But in general, the purpose of optimization is to find or 'fit' an equation that predicts, groups a dataset, or classifies as accurately as feasible. Doing this imposes the introduction of two concepts, namely:

a. the loss functions.
b. the learner or heuristic function.

The researcher may describe the learning technique as a heuristic if optimization is the approach for learning. Hyperparameters are a specific sort of parameter that the heuristic is granted control over. Hyperparameters calculate and optimize the used model by calculate or decide its own model parameters which are hidden during building or implementing a technique at the start and will not be modified during training. The method of trial and error or inference is used, or the method used may be a mixture of both. The models are expected to grow to the point where these parameters tend to explode due to the large data sets and the complexity of feature spaces [39].

A data-fitting function requires an iterative strategy to handle some minima (local or minimum) as an attempt to achieve converge, this strategy is called The Stochastic Gradient Descent (SGD) [39]. To compute a function's gradient, it uses two primary parameters: η which stands for learning rate or step size and w which stands for weight. SGD is a differentiable function that is expressed as a sum of gradients. Equation 3.1 is used to calculate it, where $Q(w)$ is the function to be minimized that what $Q_i\ (w)$ is. As a result, the i^{th} example of loss of the sample. The SGD generates a new weight after each iteration in its process. The key is balancing; the SGD must be high enough to move out of "bad" minimums while remaining modest enough not to bounce out of "good" minimums. Other optimizers use extra factors such as momentum to regulate learning behavior and tackle difficulties that SGD suffers with, such as saddle points.

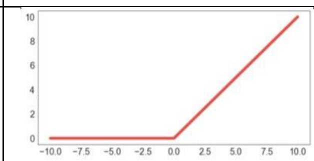
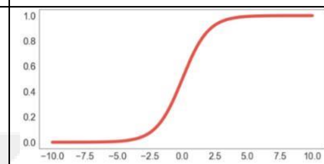
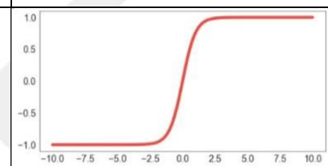
$$w := w - \eta \nabla Q(W) = W - \frac{\eta}{n} \sum_{i=1}^{n} \nabla Q_i(w) \tag{3.1}$$

Equation 3.1: Is known as SGD algorithm [39].

To drive the optimizer towards these local minima, A loss function is used to construct a "guide" [40]. They don't so much take the optimizer out of a minimum as they try to get them to trend toward the minima. Loss can be imposed at any point throughout a batch, at the conclusion of an epoch, or an iteration. The heuristic can be determined by defining the loss. It can assess how learning is enhancing or lowering overall outcomes by using loss. It also explains overfitting, which occurs when a learning process goes in the incorrect direction or excessively. The values are weighted by the model and molds the output as a form from a node or neuron via a function's activation of post neural network as a primary function of the loss function of the learning method [39]. A neuron can be activated using a variety of activation functions.

There is a cost associated with selecting one approach over another. The tasks of the loss function revolve around detecting the course of things during learning, as well as knowing whether the learning went in the wrong direction by the concept of overfitting, in addition to evaluating the learning work in terms of raising or lowering the overall results, all by trying to influence the optimizer after comparing the temporary results with "A guide" is constructed from the loss function, whereby this effect is attempted at the conclusion of an epoch or during a batch or during an iteration, whereby this optimization process is called *Stochastic Optimizers*. The activation function is used to produce weighted values that form the output from a neuron or node to some desired form, where the loss is fed to a neural network or other learning methods by the model. The researcher reviews three types of activation functions in Table 3.2 by looking at the graphics resulting from the equations, note that the wave caused by the Rectified Linear Unit (ReLU) equation was able to converge faster than the waves resulting from the rest of the activation functions [40]. After each neuron has been activated, a single layer or an entire network has the process of applying a loss function to it. Loss at the conclusion of any training session indicates an improvement in performance, most commonly accurate, indicating that the heuristic has learned some aspect of the input [41].

Table 3.2: Artificial neural network activation functions that are often used [36].

activations	functions	Plots
ReLU	$f(X) = \max\,(0, X)$	
Sigmoid	$\dfrac{1}{1+e^{-x}}$	
Hyperbolic Tangent	$\dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	

The researcher notes that it is not possible to be certain that the model has been fitted. Realizing a model has overfitted without a validation or holdout set of data, which makes Overfitting is an issue, then the optimizer reducing the loss function on the training data after switched from fitting to feature information, where the optimizer plots heuristic behavior by switching from fitting to a function and instead attempting to memorize the data. The validation data set within the model can facilitate the over-optimized identification of the learning function, but it is the method or method of its implementation that defines the overfitting problem. In other words, when the model has been trained for a very long time or when the data is suitable as possible, the problem of overfitting is obvious, and the problem of overfitting can be avoided by losing some of the knowledge gained throughout an epoch, lowering the learning method's complexity, or lowering the amount of time spent training, wherein scholar M. Telahun argued "using the wrong quantum state preparation techniques will lead to underfitting." in his thesis on the use of TensorFlow quantum [39].

3.3 QUANTUM COMPUTING

The history of quantum computation is complex and long enough to reach the current results, as there are conflicting opinions about the essence of the basics of this great field and what can provide a qualitative leap in the future of humanity and may change many previous concepts of other sciences. In 1948, at the Pocono Conference, with "Alternative Formulation of Quantum

Electrodynamics." As a title , Feynman presented his findings and the unfamiliar his diagrams to his colleagues [25]. Paul Benioff began in the 1970s investigating quantum computing's theoretical viability when he described his research described a quantum mechanical model of Turing Machines that came to light in the 1980s. In addition, Yang-Mills theory, quantum information, and mirror symmetry are among the topics on which Yuri Manin has published, Deutsch and Jozsa developed the first algorithms as a result of their contributions after several years. In 1994, Peter Shor's technique sparked interest in quantum computing by using the principle of integer factorization and discrete logarithms. What has threatened the protection of communications and data storage is the emergence of the so-called Short's Factoring algorithm that can speed up cryptanalysis techniques. And certainly, the emerging technologies face new challenges during the physical implementation of quantum computers, including the loss of information to the environment quickly, compared to quantum gates under a phenomenon called decoherence. Which in turn restricts the work of some algorithms.

Taking advantage of the properties of some metals, bringing quantum computers closer to reality, Resonant circuits are made from superconducting metals with coherent lifetimes up to millisecond-long, but the noise remains an inherent feature of quantum computers. Using certain algorithms that can be run with existing technology via usage of Noisy Intermediate Scale Quantum Devices (NISQ), for this current technology and hardware development stage, algorithms and simulations are created. Quantum computers are capable of tracking and describing qubits compared to their classical counterparts, in which quantum comprehension would require a very large set of classical numbers. The postulates of quantum mechanics are algebraic in nature, implying that there is an inherent relationship between algebraic operations and quantum computation.

In the 1990s, engineers and scientists devised the quantum circuits based on the concept of the quantum binary bit qubits to raise the exponential speed-up techniques in Principal Component Analysis (PCA), Support Vector Machines (SVM), and K-means Clustering, which depends in its work on data processing and machine learning methods using NISQ devices that lack practical realism through application development and its significance with high and diverse optimizations Resulting in multiple advances in current Quantum Computing. Quantum bits or Qubits are specified to represent the state $|\psi\rangle$ by probability amplitudes' set α_n wherein n is basis states' number as shown in Equation 3.2 which generalizes the qubits, while the constraints are illustrated in Equation 3.3 [39].

$$|\psi\rangle = \alpha_0|0 \dots 00\rangle + \alpha_1|0 \dots 01\rangle + \cdots \alpha_n|1 \dots 1\rangle \tag{3.2}$$

Equation 3.2: Representing Quantum bits [39].

$$\boldsymbol{|\psi\rangle = \alpha_0|0 \dots 00\rangle + \alpha_1|0 \dots 01\rangle + \cdots \alpha_n|1 \dots 1\rangle} \tag{3.3}$$

$$\sum_{i=0}^{n} |x_i|^2 = 1$$

Equation 3.3: Constraints on Quantum bits [39].

The classic computer has practically proven its effectiveness in solving a huge number of tasks and with endless accuracy in the results, but despite the good performance of the classic computer, the problem of speed in obtaining results remains a factor that arouses the curiosity of many researchers and computer designers. This is what quantum physics promises, which is trying hard to give a character that is summarized by exploiting the physical properties of some materials used in the manufacture of computer devices and by introducing designs for the circuits that make up these devices characterized by the application of modern quantum techniques. The application focuses on how to represent the classical state into a quantum state by creating or programming circuits of a quantum form. Since the representation of a small number of classical bits needs a very large space when converting it into equivalent qubits. Practically, the role of a classical computer is not eliminated from performing tasks, as the skill of classical algorithms has a presence role in dealing with a quantum computer. The work of both classical and quantum devices can be considered joint and complementary to each other. Quantum devices deal with complex mathematical operations that would be almost impossible to deal with by classical devices. For example, the handling of Hilbert space by quantum devices. A Hilbert space is an extension of a vector space as a complete space with an internal product structure [39]. It can be real or complex. Classical devices can treat individual values as a unit of arithmetic using Boolean algebra that operates on logic gates, in contrast to quantum devices in which vector space is the basic unit using linear algebra in quantum gates similar to matrices. Where a requirement to achieve quantum computation is to perform base change generalizations and linear operations and this is exactly what Hilbert space allows. To satisfy Hilbert space the following characteristics must be present:

i. Any complex numbers a and b in its first argument, the inner product is linear.

$$\langle ax_1 + bx_2, y\rangle = a\langle x_1, y\rangle + b\langle x_2, y\rangle \tag{3.4}$$

Equation 3.4: The inner product is linear [39].

ii. the swapped parts' inner product has complex conjugate being the same for a pair of elements [39].

$$\langle x, y\rangle = \overline{\langle x, y\rangle} \tag{3.5}$$

Equation 3.5: The complex conjugate [39].

iii. Positive-definite is the additive manufacturing technology of an element when works with itself [39].

$$\begin{cases} \langle x, x\rangle > 0 & x \neq 0 \\ \langle x, x\rangle = 0 & x = 0 \end{cases} \tag{3.6}$$

Equation 3.6: The Positive-definite [39].

The field of building quantum devices is still young, and Noisy Intermediate Scale Quantum (NISQ) gadgets currently utilized in preparing, test, and develop quantum algorithms. One of the challenges facing this generation of quantum devices is stochastic noise, which makes them prone to error. Stochastic noise is a non-ideal condition that requires work to increase the accuracy of quantum systems through a research field called quantum error correction. It is worth noting that when creating quantum devices with a process called lithographic, a junction is formed consisting of two superconducting separated metals called Josephson junction formed between the qubits. These minerals are separated by an insulating barrier, also known as quantum tunneling. The following figure shows a different number of qubits and topological connectivity for IBM quantum devices, and this difference in numbers is not directly related to their computational capabilities but rather to the term quantum volume. The term quantum volume depends on many factors including $CNOT$ error, U gate error, crosstalk, topological connectivity, circuit depth before the errors' level, and a number of qubits.

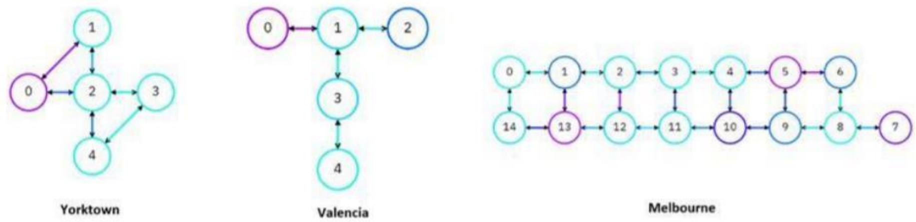


Figure 3.3: Three topologies for IBM quantum devices, courtesy of IBM Q Experience. The error rates of connections and qubits are shown by their color. Higher mistake rates are indicated by darker hues. The devices are reset on a regular basis, which affects the error rates for the better or for the worse. In the IBM Q Experience program, error rates for basic gates are also presented [39].

State changes are generated by applying the unitary gate using an arbitrary change that aims to transform a basic data on the computational basis of a qubit. For instance, Figure 3.4 shows a unitary gate or U gate on data set $\{x, y, z\}$ as inputs to obtain a new set of transformed states as outputs.

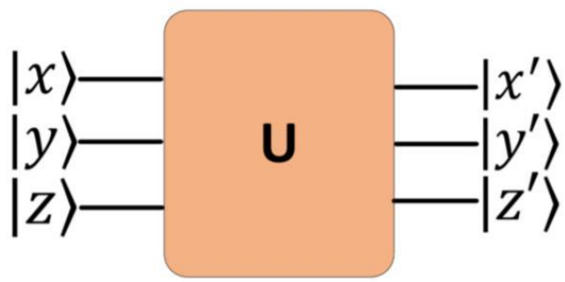


Figure 3.4: Creating state changes by applying Unitary gate[39].

3.4 MACHINE LEARNING WITH QUANTUM-ASSISTANCE

In order to increase the learning of classical methods, quantum components can be added to the basic data processing process that would achieve similar or better results using less information, reveal more appropriate patterns for quantum information, or accelerate learning to solve classical problems using quantum devices. As shown in Figure 3.5, Quantum-assisted machine learning provides four types of approaches, the data archetype is expressed as the first symbol while the computing device is expressed as the second symbol, where the two letters are combined together to express the data archetype and computing device type sequentially.

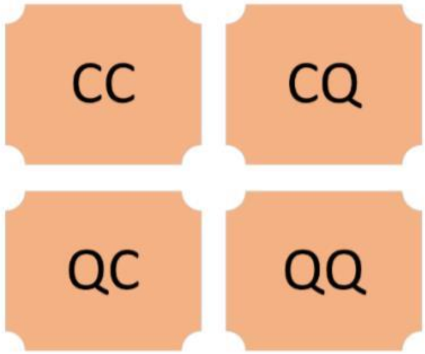


Figure 3.5: Data and system act relationship quadrant map, each of "C", "Q" stand for classical as well as quantum. The data source is represented by the first character, while the device is represented by the second. On a quantum gadget, CQ is read classical data [39].

In the NISQ era, many models emerged to overcome the issue of noise within the development of quantum machine learning computing (QML), and some of these models dealt with a methodology that applies the concept of "shots" as the basis of a working principle like Quantum Convolutional Neural Network (QCNN) model that applies iterative experiments to the probability distribution of outcomes Depending on the largest experiment value, some quantum values can be fed as optimized quantum parameters as input parameters for the next circuit, and so on, thus making this iteration of QML have a lot of the circuit's complexity and the likelihood of receiving a noisy expected outcome [39].

3.5 QUANTUM CLASSIFIERS' PARAMETRIC AND QUANTUM GATES

The researcher presents a quantum algorithm that is destined to work on quantum devices and take advantage of the characteristics of conventional algorithms and at the same time it has the ability to treat noisy devices that are short-depth circuits and works by using error-mitigation strategies to lessen the impact of decoherence. So as to make quantum circuits as simple as possible, the variational quantum classifier VQC algorithm similar to SGD algorithm is applied in its iterative way and the value of the largest output is taken to deal with the error correction issue as an attempt in order to enhance the modeling and lower the error rate for the noisy data throughout the process of transforming classical data to its quantum equivalent, amplitude encoding is used to prepare states. Figure 3.6 shows the feature map using the VQC algorithm architecture for the variational circuit's optimization.

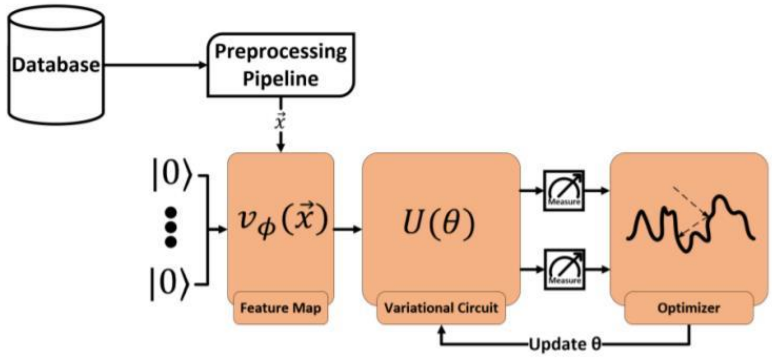


Figure 3.6: A VQC model's pipeline. After preprocessing the data for robust learning in Hilbert spaces, the feature map $\boldsymbol{v_{\phi}(\vec{x})}$ is applied to it. The variational quantum circuit is then employed after that. The circuit's output is sent into a typical device's optimizer, which changes the parameter θ [39].

It has conclusively been shown that in order to perform optimization on the variational circuit, it is necessary to first apply the feature map, where the classical data is converted to its quantum state $|\psi(x_i)\rangle$ within "black-box" encoding work to find transformations to the ground state $|0\rangle^n$ [35], Where classical data is represented in a quantum system with higher dimensions within the Hilbert space [42], [43] , as shown in Equation 3.7 and Equation 3.8.

$$v_{\phi}(\vec{x}) = |\Phi(\vec{x})\rangle = U_{\Phi(\vec{x})} H^{\otimes n} U_{\Phi(\vec{x})} H^{\otimes n} |0\rangle^{\otimes n} \tag{3.7}$$

Equation 3.7: Equation the feature map to find $\boldsymbol{H}$ known as Hadamard gate [42], [43].

$$U_{\Phi(\vec{x})} = exp\left(i \sum_{s \subseteq [n]} \phi s(\vec{x}) \prod_{i \in x} z_i \right) \tag{3.8}$$

Equation 3.8: Equation a diagonal gate in the Pauli $\boldsymbol{Z}$ – basis [42], [43].

Quantum bits, also known as qubits, are quantum counterparts of classical bits. In a two-dimensional Hilbert space $\mathcal{H}_2$, qubit is unit vector with an orthonormal basis, denoted by basis $\{|0\rangle, |1\rangle\}$, has been fixed, which can be superimposed when complex numbers α and β, then can be represented as $\alpha|0\rangle + \beta|1\rangle = 1$, this superposition can be measured as the isotope of classic 0 is $|\alpha|^2$ while the isotope of classic 1 is $|\beta|^2$. Every n-qubit quantum gate is unitary matrix U defined over Hilbert space n-qubit that represents linear change, for which the conjugate transpose is $U^{\dagger}$ then $UU^{\dagger} = I$. There are many types of unitary gates, including:

i. The Pauli set of single-qubit gates is shown in Equation 3.9:

$$\sigma_0 = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \sigma_1 = X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \sigma_2 = Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \sigma_3 = Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (3.9)$$

Equation 3.9: Elements as single-qubit gates of the Pauli set [44].

ii. The Bloch sphere is a unitary gate around rotation operators of the axis's (x, y, z) [44] as shown in Equation 3.10:

$$\mathcal{R}_x(\alpha) = \begin{bmatrix} \cos\frac{\alpha}{2} & -i\sin\frac{\alpha}{2} \\ -i\sin\frac{\alpha}{2} & \cos\frac{\alpha}{2} \end{bmatrix}, \mathcal{R}_y(\alpha) = \begin{bmatrix} \cos\frac{\alpha}{2} & -\sin\frac{\alpha}{2} \\ \sin\frac{\alpha}{2} & \cos\frac{\alpha}{2} \end{bmatrix}, \mathcal{R}_z(\alpha) = \begin{bmatrix} e^{-i\frac{\alpha}{2}} & 0 \\ 0 & e^{i\frac{\alpha}{2}} \end{bmatrix} \quad (3.10)$$

Equation 3.10: The Bloch sphere with rotation operators as single qubit unitary gates around $\boldsymbol{x}, \boldsymbol{y}$ and $\boldsymbol{z}$ axis with the angle $\boldsymbol{\alpha}$ [45], [46].

iii. Hadamard, H, and T gates are shown in Equation 3.11:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, T = \begin{bmatrix} e^{i\frac{\pi}{8}} & 0 \\ 0 & e^{-i\frac{\pi}{8}} \end{bmatrix} \quad (3.11)$$

Equation 3.11: Single-qubit gates ($\boldsymbol{H}, \boldsymbol{T}$) called Hadamard [46].

$C^k\ U$ gates have one target and k control qubits, a controlled-U gate can be considered as a gate operating on two qubits are control and target qubits assuming that a gate works on a single qubit is known as U based on it only if a control qubit value equal $|1\rangle$ then U is applied to a qubit target. Qubits and flow time travel from left to right through quantum wires that connect quantum gates together to form the quantum circuits. The quantum circuit's unitary matrix is calculated by taking the tensor product or dot product of quantum gates' unitary matrices, so the effect on those gates is calculated. The neighboring gates that work on distinct qubits' subsets may be utilized simultaneously time, and the tensor product is used to calculate the total net impact as shown in Equation 3.12:

$$A \otimes B = \begin{bmatrix} a_{11}B, & a_{12}B & \cdots & a_{1n}B \\ a_{21}B, & a_{22}B & & a_{2n}B \\ \vdots & & \ddots & \vdots \\ a_{m1}B, & a_{m2}B & \cdots & a_{mn}B \end{bmatrix} \tag{3.12}$$

Equation 3.12: The tensor product (Kronecker product) when $\boldsymbol{A}$ is $\boldsymbol{m} \times \boldsymbol{n}$ matrix, $\boldsymbol{B}$ is $\boldsymbol{p} \times \boldsymbol{q}$ matrix [46].

3.6 THE SIMULATING MODEL TO COMPUTATIONAL QUANTUM CIRCUIT MODEL

3.6.1 Quantum Computation's Cluster-State Model

Cluster state is a fixed quantum state over wherein a set of single-qubit measurements for coherent quantum information processing [45], such that a classical computer may efficiently mimic measurements on every one-dimensional quantum state that has been linearly prepared as well as any naturally occurring physical sequence system cannot have the cluster state as its precise ground state and this is what characterizes the cluster state model. What prompted researchers to develop a thorough knowledge of simulation from a physics standpoint is that a physical system can be simulated by another physical system, so it is believed that hypothetical quantum computers would be able to simulate any other physical system efficiently in the near future. [42]. As usual, physical resources remain the focus of many researchers, which requires a sufficient universal for quantum computation used in building a quantum computer. Models for quantum computation are measurement based where non-unitary quantum measurements are used in all basic dynamical procedures for this model. It's worth noting that it can mimic any quantum dynamics, including unitary dynamics. The so-called one-way quantum computer, or cluster-state model, suggested by Raussendorf and Briegel [46], is a family of measurement-based quantum computing models. The cluster-state model has a structured form which are still being deciphered, but it varies significantly from the traditional unitary computing quantum paradigm. These distinctions have resulted in new insights into quantum computing complexity [47] as well as drastic simplifications in experimental quantum computation approaches [48]–[51] suggested and developed an alternative technique to measurement-based quantum computing [52], [53]. The research on measurement-based quantum computing, in general, has exploded in recent years. A quantum-mechanical model of computing may be built in many various methodologies a priori. All the physically viable quantum computing models that have been presented so far have shown to be computationally similar. That is, any of these seemingly disparate models of computing can effectively replicate the others, as a result, they don't differ in terms of the kinds of computational issues they potentially handle. The most well-known *quantum circuit model* for quantum computing is a modification around *classical circuit model* whose basis of operation is “OR” and “AND” as Boolean logical operations. The operation of the quantum circuit model can be summarized like a brief overview about conventional unitary quantum circuits quantum computing model:

The quantum circuit model is considered to be quantum computing's conventional model. The following example on a quantum circuit in Figure 3.7 illustrates the essential features of the quantum circuit model:

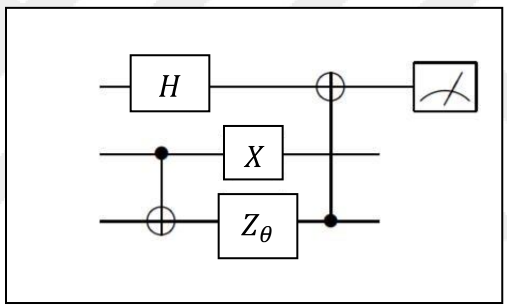


Figure 3.7: The essential features of the quantum circuit model [45].

The horizontal lines represent quantum wires, the (orthogonal) computational basis states $|0\rangle$ and $|1\rangle$ span wo-dimensional quantum mechanical systems. The left-to-right travel of a wire represents progress through time rather than movement of the qubit across space. The entire $|0\rangle$ state, $|0\rangle^{\otimes n}$ n, is often used as the qubits' starting state. The qubits are processed via a sequence of one-qubit or two-qubit quantum gates. These quantum gates are one- or two-qubit unitary operations that change their states. A single-qubit gate, for instance, seems to be the Hadamard gate as an illustration in Figure 3.8.

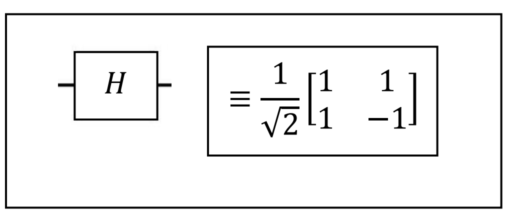


Figure 3.8: Hadamard gate [45].

The unitary matrix encoding regarding the $|0\rangle, |1\rangle$ basis illustrates the Hadamard gate's action is shown on the right which takes an input $|0\rangle$ then converts into $(|0\rangle, |1\rangle)/\sqrt{2}$. Rotations about axes X, Y, Z are also significant single-qubit gates as sown in Figure 3.9:

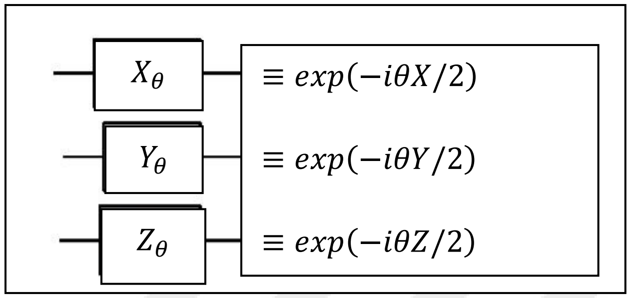


Figure 3.9: Rotations about axes X, Y, Z are also significant single-qubit gates [45].

wherein X, Y, Z are Pauli's shorthand notations $\sigma_x, \sigma_y, and\ \sigma_z$ matrices, respectively. Perhaps the most extensively utilized two-qubit gate is the controlled-not gate as illustration in Figure 3.10:

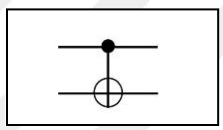


Figure 3.10: The controlled-not gate [45].

The control qubit is on top, while the target qubit is on bottom. Because the computational basis has the Controlled-not effects wherein $\oplus$ is addition mod 2 to convert $|x, y\rangle$ to $|x, y \oplus x\rangle$, these names are employed. That is, the target is flipped if the control is set to 1 otherwise the target is stays untouched, so the control qubit is unaffected. the more controlled-unitary gates have a subset so-called Controlled-not gates as sown in Figure 3.11:

Figure 3.11: Controlled-unitary gates [45].

The control qubit is used in both circumstances, whereas the target qubit is used in the other. U is applied to the target qubit in the first circuit if the control qubit is set to 1, but U is applied to the target qubit in the second circuit if the control qubit is set to 0. Aside from the controlled-not, the controlled-phase gate is another kind of controlled-unitary gate that is used.

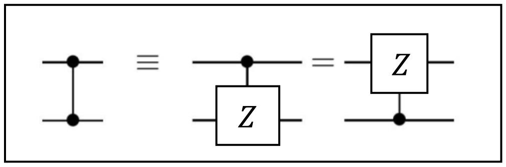


Figure 3.12: The controlled-phase gate [45].

As shown in Figure 3.12, $|x, y\rangle \rightarrow -1^{xy}|x, y\rangle$ is the computational basis's controlled-PHASE activity. Total impact of the gates in a circuit looks like on the *n* input qubits that (1) is a joint unitary transformation because the individual gates are unitary. A question that draws attention is which unitary operations are possible to synthesize in this manner? It turns out that if the accessible gates include a two-qubit entangling gate at the very least and all single-qubit gates, the qubits may then be utilized to synthesize any unitary operation, in other words the gate set is universal, such as the controlled-PHASE or Controlled-not [54]. Finding tiny quantum circuits that synthesize suitable unitary operations is the difficulty of quantum computing. The number of gates needed to synthesize a generic unitary U over *n*-qubits increases exponentially on *n* [55]. Unitary operations may be synthesized by finding an exponential scale or by using a few gates that scales polynomial in (*n*) which is the most desirable family. Many simple forms of the quantum circuit concept are often utilized in practice. These are some of them:

i. Accepting any tensor product of single-qubit values as the input state, $|\psi_1\rangle \otimes |\psi_2\rangle \otimes ... \otimes |\psi_n\rangle$. A sequence of bits made up of zeros and ones a sequence of bits made up of zeros and ones the state of the sub-qubits in the quantum circuit is read by measuring them in their respective computational bases. wherever it is a measurement in the computational base is indicated by the meter ending the top wire.

ii. Measurements and feedforward of measurement findings are possible throughout the computation, allowing subsequent actions to be influenced by the results of previous measurements none of these adjustments increase the quantum circuit model's processing capability, but they do make it easier to work with.

iii. Measurements on any single-qubit orthonormal basis are allowed Because a computational basis measurement may be accomplished after the single-qubit unitary operation.

3.6.2 A Cluster-State Computation

Quantum circuits effectively simulated using cluster-states model [46], [56]. A cluster-state is a quantum state with a large number of entangled qubits are set up and the cluster is processed via a multi-qubit adaptive series of measurements and then reading the outcome of the calculation from the remaining qubits. The idea of working the cluster-state can be summarized as follows:

it may be established a corresponding n-qubit cluster state for any graph G with n vertices by after connecting a matching qubit to each vertex, the qubits are prepared using a graph-dependent technique. For an instance, Figure 3.13 depicts a six-qubit cluster state.

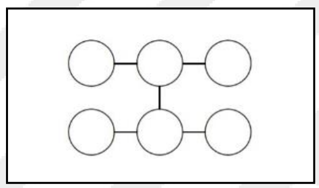


Figure 3.13: Six-qubit cluster state [45].

The following preparation technique may be used to determine a graph's cluster state:

a. Assemble each of the state's n qubits $|+\rangle \equiv (|0\rangle + |1\rangle)/\sqrt{2}$.

b. Between qubits with connected graph vertices, controlled-PHASE gates are employed.

Because the controlled-PHASE gates commute, there is no need to specify the order in which they are applied. In terms of applying quantum gates the cluster's construction allows for an employment of classical devices as models of quantum devices, but clusters may also be prepared via measurements, making the cluster-state model a true measurement model of quantum computing. Originally, A set of states connected is indicated by "graph state" which is more broad graphs G. The development of quantum computing techniques based on the principles of Raussendorf and Briegel [57], they demonstrated that this class of states may be employed as a quantum computer environment when they coined the word "cluster state" but using different graphs when the graph G could be two-dimensional square lattice [46]. The next calculation phase executing series of processing single-qubit measurements over cluster state once it has been constructed. Because the measurement foundation used may be influenced by the results of previous measurements, feedforward of classical measurement results is permitted. Consequently, the measurement findings might be handled by a traditional computer to help with the feedforward process. For more efficiency in the cluster-state computation, the classical calculation must be of polynomial size. This is done in one of two ways:

a. Make the computation's output a classical bit string After the processing measurements are complete, a series of single-qubit measurements (a set of read-out measurements) are applied to the qubits that remain.

b. Consider that computation after the processing measurements sequence has ended, it has the quantum state of the qubits that remains, i.e., as a quantum state as output.

For instance, shown in Figure 3.14:

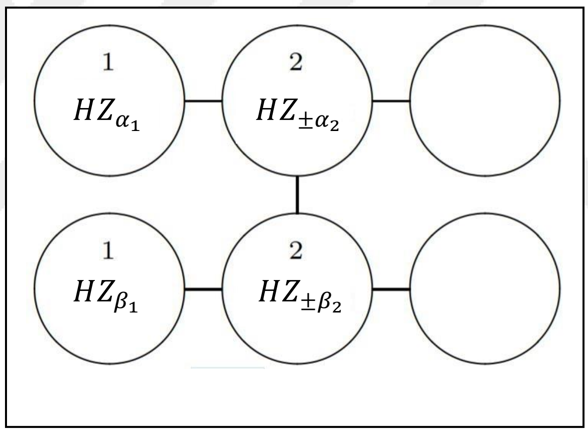


Figure 3.14: A quantum state as output [45].

After the processing measurements are completed, the computation's output remains unlabeled qubits, while processing measurements actually occur on labeled qubits. A single-qubit unitary $U = HZ_{\pm\alpha_j}, HZ_{\pm\beta_j}$ and a positive integer *n* are used to designate the qubits. Then, the qubits can be measured in simultaneously or in either order, if they have the same label, The measurement results are determined by the processing measurements' time-ordering which will be sent forward to control subsequent measurement rules. The U label designates the measurement basis for the qubit, a computational basis measurement follow the suggested unitary U rotation. In the basis $\{U^{\dagger}|0\rangle, U^{\dagger}|1\rangle\}$, a single-qubit measurement is conducted. The notation $HZ_{\pm\beta_2}$and $HZ_{\pm\alpha_2}$denotes that that the results of earlier measurements impact the sign choice in a manner that must be determined separately.

3.6.3 Using The Cluster-State Model's Computation To Simulate Quantum Circuits

A basic circuit identity (one-bit teleportation) utilizes for imitate quantum circuits using cluster-state computation [58].

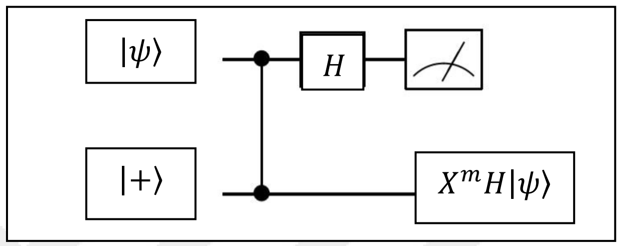


Figure 3.15: One-bit teleportation [45].

What Figure 3.15 is that by expanding $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, this identity may be verified (wherein the first qubit's computational basis measurement results be m as 0 or 1) by the gate definitions given earlier, so that the state after the Hadamard gates and controlled-PHASE is $\alpha|++\rangle + \beta|--\rangle$. As a consequence of which for this state, the re-expressed equation may be $(|0\rangle \otimes H|\psi\rangle + |1\rangle \otimes XH|\psi\rangle)/\sqrt{2}$. Thus, the identity generalized to the following identity is illustrated in Figure 3.16:

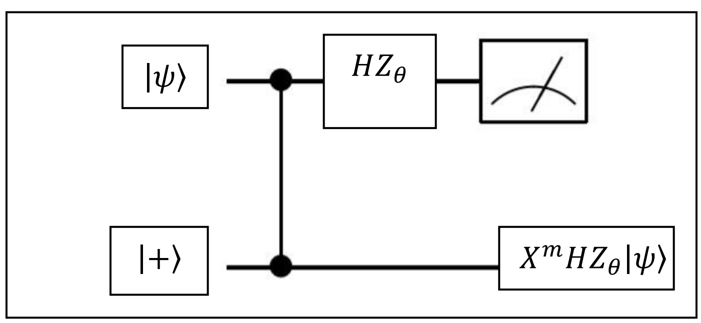


Figure 3.16: The identity generalized to the following identity [45].

The evidence lies in the fact that Z_θ uses the controlled-PHASE gate to commute, therefore the circuit's output is the same as it would have been if $Z_\theta|\psi\rangle$ had been used instead of $|\psi\rangle$ in Equation of Figure 3.16. No quantum information is lost even when the first qubit is measured because regardless of the measurement result, the second qubit's posterior state is related to the original input using known unitary transformation $|\psi\rangle$. The two qubits' current states are swapped by a swap gate instead of the controlled-PHASE gate. Before measurement, all quantum information will be transmitted to second qubit from first qubit since no quantum data has been destroyed as a result of a measurement on the first qubit. This is contrary to the norm, meaning It is possible to modify the measurement basis for the first qubit withou' losing any quantum information by changing the unitary transformation applied to the second qubit, such as the EPR phenomenon has been generalized in this way [59], as well as a quantum error-correcting code example The above equation may be used to demonstrate how cluster-state computing can be used to mimic quantum circuits. The next kind of single-qubit circuit may be simulated as shown in Figure 3.17:

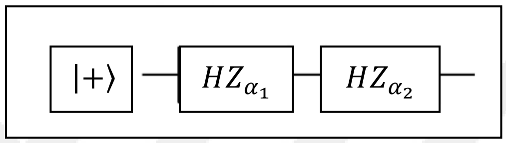


Figure 3.17: Kind of single-qubit circuit [45].

It's worth noting that assuming that single-qubit gates are HZ_α type and qubit begins in the $|+\rangle$ state. These assumptions have no bearing on the system's generality because it's self-evident that any single-qubit circuit capable of simulating these operations may be emulated. For the previous circuit the cluster-state computation will be as sown in Figure 3.18:

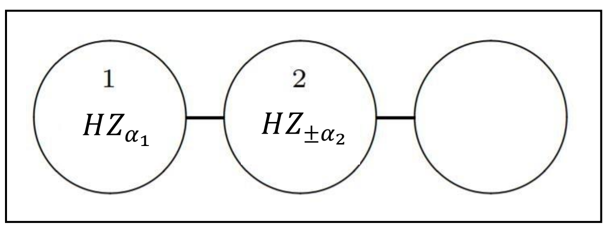


Figure 3.18: Cluster-state computation [45].

The result of this cluster-state computation is the same as the output of the quantum circuit described below which clarified in Figure 3.19, according to the definition:

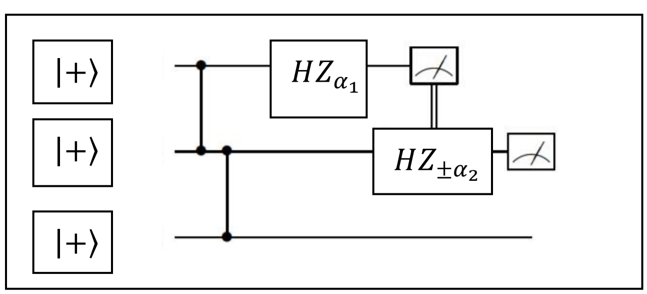


Figure 3.19: The result of this cluster-state computation [45].

The Figure 3.20 shows that alternatively, after operations on the first qubit measurement is complete its can be start on the second and third qubits:

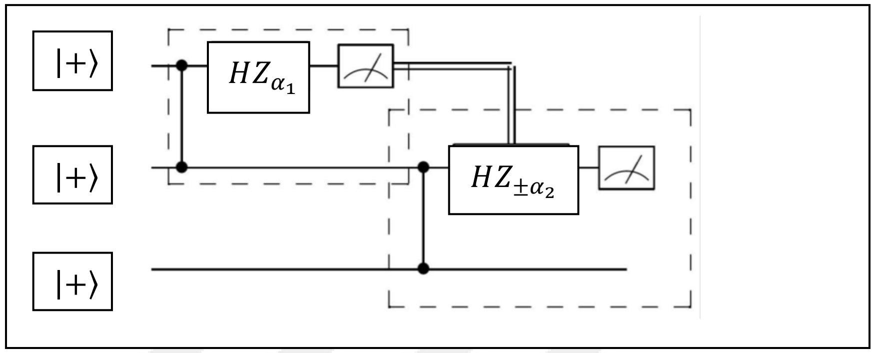


Figure 3.20: Operations' sequence on qubits measurement [45].

If m_1 is the output of the first qubit's measurements and the output of the second qubit' measurements is m_2, then output may be determined by the circuit $X^{m_2}HZ_{\pm\alpha_2}X^{m_1}HZ_{\alpha_1}|+\rangle$ as a result, the feedforward may be utilized to pick the sign of $\pm\alpha_2$, resulting in $Z_{\pm\alpha_2}X^{m_1} = X^{m_1}Z_{\alpha_2}$. Consider $HX^{m_1} = Z^{m_1}H$; the output may be recast as $X^{m_2}Z^{m_1}HZ_{\alpha_2}HZ_{\alpha_1}|+\rangle$, making the known Pauli matrix $X^{m_2}Z^{m_1}$ the same as a single-qubit quantum circuit's output, with HZ_α gates, this example may be simply adapted to massive single-qubit circuits. While as seen in Figure 3.21, multi-qubit quantum circuits may be simulated by cluster-state computation:

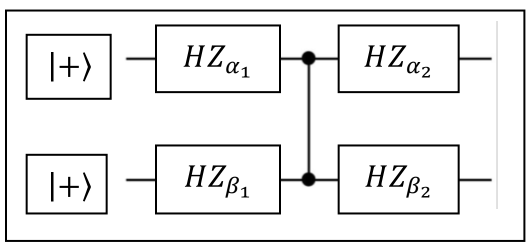


Figure 3.21: Simulating multi-qubit quantum circuits [45].

To summarize, it became evident that effectively simulate any quantum circuit utilize computing cluster-state model with all $|+\rangle$ states as gates and inputs with either *HZ* gates or controlled-phase gates. The cluster-state model as resources is ubiquitous for quantum computing so efficiently simulate any quantum circuit. In contrast, In the quantum circuit model, any cluster-state computation possible correctly simulated, meaning that two models are equivalent in terms of computational complexity.

3.6.4 The Cluster-State Model's Properties

For quantum computing, cluster states are valuable substrates. Where the spatial dimension is an important factor. Linearly assembled quantum states (a state $|\psi\rangle$) created by one-dimension possible effectively simulated using classical devices, so they're considered useless for quantum computing. For instance, as a linearly assembled quantum state of some product starting state, quantum measurements in a succession on a series may be performed wherever the basis for subsequent measurements may be prepared, these subsequent measurements are influenced by the findings of previous measurements to make use of a circuit in the form shown in Figure 3.22:

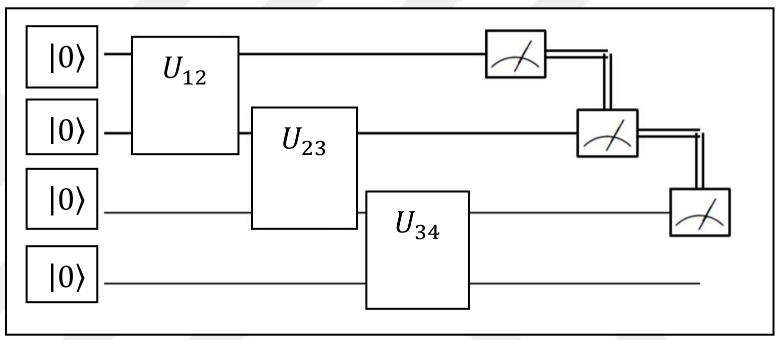


Figure 3.22: Subsequent measurements are influenced by the findings of previous measurements [45].

The computational basis, on the other hand, is comparable to the circuit shown in Figure 3.23:

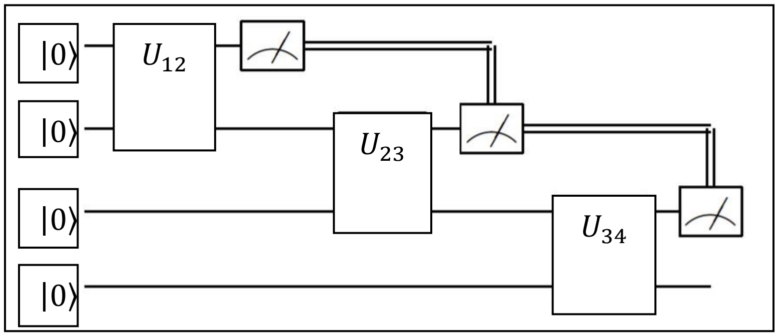


Figure 3.23: The computational basis [45].

The circuit in the above figure possibly simulated using classical computer, it means, a classical simulation of the calculations is carried out in order to obtain an accurate operational result. U_{12} is applied to $|0\rangle$ to calculate the probabilities of the measurement on the first qubit, then generate a posterior state (as inputs for U_{23}) after adopted the currently samples distribution for the subsequent qubit, after that, the process of calculating probabilities, sampling, and computing posterior states is repeated. It is necessary to execute calculations to $O(\log\left(\frac{1}{n}\right))$ bits of accuracy in order for the final result to be exact for an efficient classical simulation. To achieve this taking into consideration the necessity of multiplying the integers of the floating point (c), the computation process requires performing $O\ (n\ \log^{c}\left(\frac{1}{n}\right))$ operations.

To further clarify and prove that measure the qubits in a different sequence perhaps led to approach has a polynomial with an n-time overhead, making it efficient, the researcher reviews the example in the Figure 3.24 that measures the qubits in a different order:

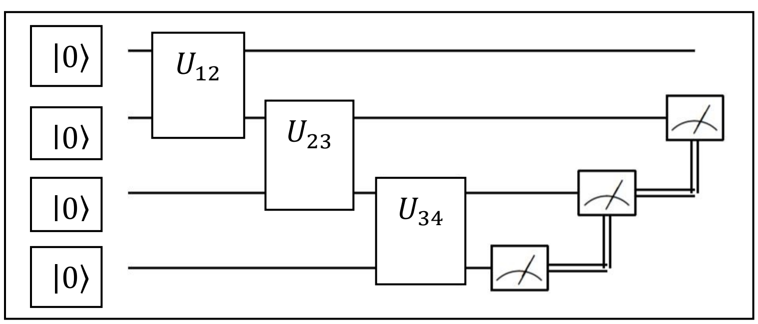


Figure 3.24: Measures the qubits in a different order [45].

With $n = 4$ qubits involved, the current state (ρ_j) is defined before the subsequent ($U_{j,j+1}$) qubit is applied but after the ($U_{j-1,j}$) qubits can be applied, the n corresponding qubits can be measured to the two possible measurement outcomes by deducing trace-decreasing operations $\mathcal{E}_{n,0}$ as well as $\mathcal{E}_{n,1}$ over qubit $n - 1$, then calculating state measurement (ρ_{n-1}) to simulating the nth qubit measurement. All of the potential measurement outputs may be simulated using a classical computer by iterating this technique. Since the computations must be accurate, it done with $O(n)$ bits of accuracy, this approach is efficient since its time overhead is polynomial in *n*. Based on the foregoing, the unitary processes at the quantum circuit model's core may be replicated using measurements on cluster states.

3.7 DEEP QCNN ARCHITECTURE WITH MERA

The Density Matrix Renormalization Group (DMRG) gives an understanding of quantum entanglement in quantum information science of one-dimensional (1D) systems, which has led to a fundamental gain in numerical simulations of multi-body quantum systems [60]–[64]. Focusing on quantum systems on a lattice, entanglement renormalization employs a quantum systems circuit for a rigorous evaluation of local expectation values with logarithmic depth and distinct causal structure, as demonstrated by MERA, on a D-dimensional lattice, the Multi-Scale Entanglement Renormalization Ansatz (MERA) may be utilized to effectively describe certain quantum many-body states [65], such as address $2D$ systems [61] and efficient algorithms to simulate time-evolution [60].

To efficiently represent the states of quantum multibody objects by these algorithms is to use a network of tensors. The tensors in the network are connected within specific structures that are determined according to the dimensions of the link system. The tensor network is Matrix Product States (MPS) if the system is one-dimensional [66], or the system may be two-dimensional and more, then the tensor network is known as Projected Entangled-Pair States (PEPS) [61], alternatively the system may have the structure of a tree, in which case the tensor network would be known as Tree Tensor Networks (TTN) [62].

MPS is illustrated in Figure 3.25:

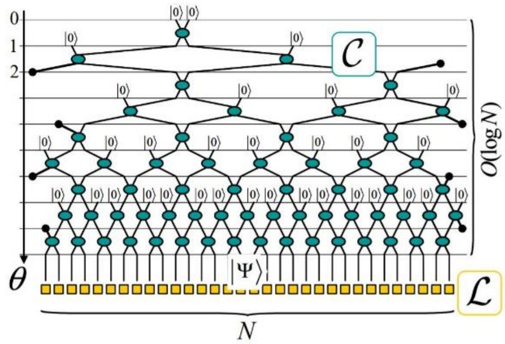


Figure 3.25: Matrix product states (MPS). In a 1D lattice $\boldsymbol{\mathcal{L}}$, the $|\mathbf{0}\rangle^{\otimes \boldsymbol{N}}$ state is transformed by a quantum circuit C containing $\mathbf{2}\,\boldsymbol{N}\,-\,\mathbf{1}$ gates into the N-site state $|\boldsymbol{\Psi}\rangle$, these layers are organized in a discrete time $\boldsymbol{\theta}$ in the form of $\boldsymbol{O}(\mathbf{log}\,\boldsymbol{N})$ layers [65].

2D PEPS in Figure 3.26:

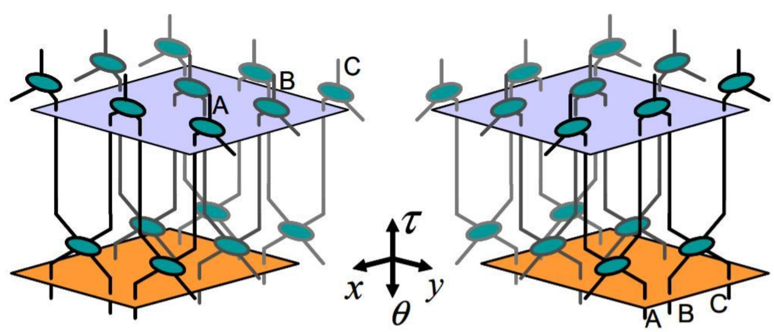


Figure 3.26: Projected Entangled-pair states (PEPS). In a 2D MERA, the width of a causal cone is bounded, layers of tensors work with $\mathbf{3}\,\times\,\mathbf{3}$ sites (top right) mapped from $\mathbf{3}\,\times\,\mathbf{3}$ sites (bottom left), these layers act along the y and x direction [65].

And TTN in Figure 3.27:

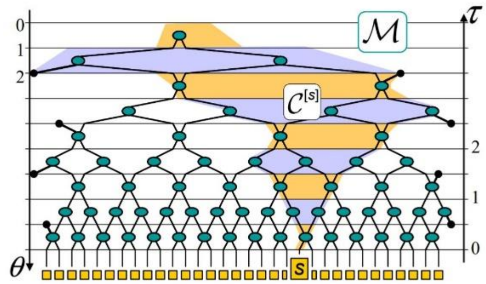


Figure 3.27: Tree Tensor Networks (TTN). $\mathcal{M}$ implements entanglement renormalization transformations when reversing the arrow of time $\boldsymbol{\theta}$ because it is bounded width in the causal cone $\boldsymbol{C}^{[s]}$ for site which makes the causal structure of a quantum circuit C is inherited by a MERA $\mathcal{M}$ [65].

Wherein the local observables' expectation value can be efficiently computed by tensor networks. A PEPS contains a much wider range of applications compared to a MPS and a TTN attached with a lower efficiency in obtaining local expectation values. This efficiency increases in a PEPS with an increase in a number of approximations. In this thesis, local expectation values can be computed exactly through MERA to encode D-dimensional lattice systems' quantum many-body states.

MERA tensors are isometric in D 1 dimension, but for extra dimension according to successive applications in entanglement renormalization, it may be believed that parameterizing diverse length scales into a system or a particular time class of quantum calculations, which is essentially a lattice coarse-graining operation, is a prerequisite for quantum computing [67].

From emerging quantum phenomena are quasi-particle excitations, quantum phase transitions, and topological order, this is what MERA attempts to describe from preserving entanglement through entanglement renormalization with the following characteristics:

a. Capacity to internalize symmetries naturally, like rescaling or translation invariance, resulting in significant benefits in processing efficiency.

b. Adaptability to both the system's underlying lattice's local and global structure.

c. intrinsic support for algebraically fading correlations and an entanglement area law.

d. The network's contractibility efficiency, resulting in an accurate evaluation of local expectation values.

In Figure 3.25, Quantum circuit C contains $2N - 1$ gates to treat state $|0\rangle^{\otimes N}$ as arranged into $O(\log N)$ layers with discrete time labels θ to convert it to the N-site state $|\Psi\rangle$ of a 1D lattice $\mathcal{L}$ [1]. Assuming that N sites have D spatial dimensions to form a square lattice $\mathcal{L}$, where a complex vector space $\mathbb{V}$ describes each site $s \in \mathcal{L}$ within finite dimension χ. Then the density matrix for site s can be produced by $\rho^{[s]} = tr_{\bar{s}}(|\Psi\rangle\langle\Psi|)$ while a pure state of lattice $\mathcal{L}$ can be denoted by $|\Psi\rangle \in \mathbb{V}^{\otimes N}$. States $|\Psi\rangle$ with depth of $\Theta \equiv 2\log_2(N) - 1$ may be produced using a certain quantum circuit C. The one-dimensional state is illustrated in Figure 3.25 where both are labeled with the same index s where each one outgoing wire of C there is a compatible site in $\mathcal{L}$ with it.

The external wiring contains the set of gates and wires known as the causal cone $C^{[s]}$ which of course may influence state $\rho^{[s]}$, each causal cone $C^{[s]}$ has bounded width, more precisely, there is the constant which is not altered by N for the number of wires in $C^{[s]}$ within a time slice $C_{\theta}^{[s]}$. a discrete time with label $\theta \in \{0, 1, \cdots, \Theta\}$. Looking closely at Figure 3.25, it can be concluded that during any time θ there are any outgoing wire involves four wires at most, which was highlighted in Figure 3.26, while Figure 3.27 shows that the causal cone of outgoing wires within a given time θ comprise at most 3D in the D-dimensional case. More generally, when there is a quantum circuit C then a tensor network $\mathcal{M}$ is compatible with it and may be considered A MERA for $|\Psi\rangle$ except for some minor changes. In the tensor network incoming wires in state $|0\rangle$ are eliminated while a tensor of $\mathcal{M}$ results from each two-body unitary gate u of C, so three types of tensors will be formed:

i. the top tensor $t = u|0\rangle|0\rangle$ of $\mathcal{M}$ is normalized to 1 and has two indices,

$$(t)_{\mu\nu} = (u)_{\mu\nu}^{\alpha\beta}\Big|_{\alpha,\beta=0}, \qquad \sum_{\mu\nu}(t^*)_{\mu\nu}(t)_{\mu\nu} = 1 \tag{3.13}$$

Equation 3.13: The top tensor of $\mathcal{M}$ with two indices.

ii. Every second row of tensors is isometries $w = u|0\rangle$,

$$(w)_{\mu\nu}^{\alpha} = (u)_{\mu\nu}^{\alpha\beta}\Big|_{\beta=0}, \qquad \sum_{\mu\nu} (\mathcal{W}^{*})_{\mu\nu}^{\alpha} (\mathcal{W})_{\mu\nu}^{\acute{\alpha}} = \delta_{\alpha\acute{\alpha}} \tag{3.14}$$

Equation 3.14: Isometries $\boldsymbol{w}$ in Every second row of TENSORS.

iii. Unitary gates u called disentanglers make up the rest of M's tensors,

$$\sum_{\mu\nu} (u^{*})_{\mu\nu}^{\alpha\beta} (u)_{\mu\nu}^{\acute{\alpha}\acute{\beta}} = \delta_{\alpha\acute{\alpha}} \delta_{\beta\acute{\beta}} \tag{3.15}$$

Equation 3.15: Disentanglers of Unitary gates $\boldsymbol{u}$ part 1.

$$\sum_{\alpha\beta} (u^{*})_{\mu\nu}^{\alpha\beta} (u)_{\acute{\mu}\acute{\nu}}^{\alpha\beta} = \delta_{\mu\acute{\mu}} \delta_{\nu\acute{\nu}} \tag{3.16}$$

Equation 3.16: Disentanglers of Unitary gates $\boldsymbol{u}$ part 2.

According to equations (Equation 3.13, Equation 3.14, Equation 3.15 and Equation 3.16), the computational space grows linearly in the form of $O(\chi^{4}\text{N})$ to store $\mathcal{M}$, the fact that each tensor depends on χ^{4} parameters at most, with a number of $2N - 1$ tensors. Thus, the tensor network within the dimensions $D + 1$ becomes a component of MERA, which describes $|\Psi\rangle$, according to certain properties that are summarized as follows:

i. Each open wire s has a restricted width causal cone $C^{[s]}$ connected with one site of the underlying lattice $\mathcal{L}$.

causing reduction in calculating cost of reduced density matrix in the case of a small number of lattices $\mathcal{L}$ sites.

ii. Equation 3.13 and Equation 3.14 or Equation 3.15 and Equation 3.16 are considered to be constraining tensors.

The spatial dimension D of $\mathcal{L}$ on which $\mathcal{P}1$ and $\mathcal{P}2$ depend as integer numbers; the lemmas will be a k-site reduced density matrix $\rho^{[s_1 \cdots s_k]}$ where they can be easily extended. At the same time, they suggest that local observables, such as two -site correlators, have an expected value.

Lemma 1: The one-site density matrix $\rho^{[s]}$ may be calculated in time $O(\chi^{\mathcal{P}_1} \log N)$ using a MERA.

Proof: The reduced density matrix σ_θ can be constructed for each time slice $C_\theta^{[s]}$ of the causal cone $C^{[s]}$. For $D = 1$, as shown in Figure 3.28, $\sigma_{\theta+1}$ may be produced from σ_θ using a polynomial cost in χ and is independent of N. Finally, remember that $\rho^{[s]} = \sigma_\Theta$ and that $\Theta = 2\log_2(N) - 1$.

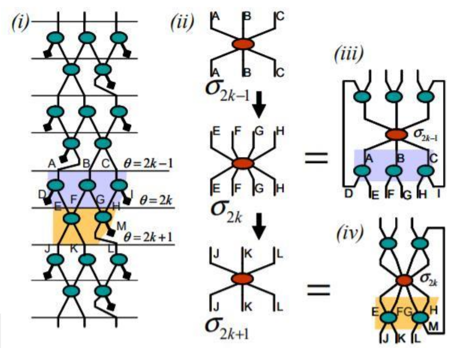


Figure 3.28: MERA structure with Lemma 1 [65]

(i) D = 1, calculate Part of the causal cone $\boldsymbol{C}^{[s]}$ of $\boldsymbol{\rho}^{[s]}$.

(ii) through simple tensor multiplications it is possible to compute σ_{2k+1} from σ_{2k-1} using (iii) and (iv), where the calculation of $\rho^{[s]}$ is very efficient.

Lemma 2: The density matrix $\rho^{s_1 s_2}$ for two sites may be calculated in $O(\chi^{\mathcal{P}_2} \log N)$ time.

Proof: As shown in Figure 3.29, regardless of N the causal cone $C^{s_1 s_2}$ has logarithmic depth and width, as well as a reduced density matrix $\sigma_{\theta+1}$ in order to time slice $C_{\theta+1}^{s_1 s_2}$ may be generated at limited cost from for $C_\theta^{s_1 s_2}$.

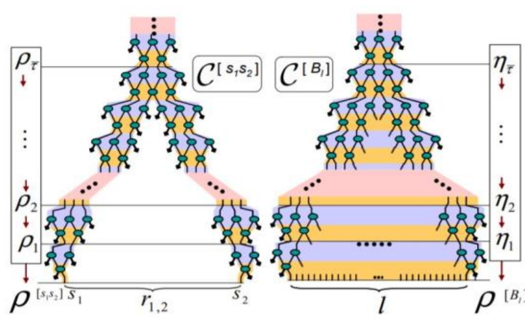


Figure 3.29: MERA structure with Lemma 2 [65]

(The left side): Causal cones $\boldsymbol{C}^{[s_1 s_2]}$ for a two-site density matrix $\boldsymbol{\rho}^{[s_1 s_2]}$.
(The right side): Causal cones $\boldsymbol{C}^{[B_l]}$ for an l-site density matrix $\boldsymbol{\rho}^{[B_l]}$.

Local observables, such as two-site correlators, can have their expected value determined efficiently from Equation 3.17

$$C_2(s_1, s_2) \equiv \langle\Psi|A^{s_1}B^{s_2}|\Psi\rangle = tr(\rho^{s_1 s_2}A^{s_1}B^{s_2}) \tag{3.17}$$

Equation 3.17: Expected value determined efficiently.

Θ layers of unitary gates are used to transform a product state $|0\rangle^{\otimes N}$ into $|\Psi\rangle$ by a quantum circuit C with the purpose of defining a MERA for $|\Psi\rangle$. The non-trivial part of circuit Cs state is represented by $|\Psi_\tau\rangle$ at time $\theta = \Theta - 2\tau$, with $|\Psi_0\rangle \equiv |\Psi\rangle$. In a more precise sense one can consider the sequence of states $\{|\Psi_0\rangle, |\Psi_0\rangle, |\Psi_0\rangle ... \}$ can interpretation of the MERA, considering the quantum evolution of C back in time. With $\mathcal{M}$ implements entanglement renormalization transformations, two layers of tensors can be applied with the aim of obtaining $|\Psi_{\tau+1}\rangle$ from $|\Psi_\tau\rangle$:

a. First layer is built up of disentanglers which make the state $|\Psi_\tau\rangle$ evolves into a less entangled case as state $|\acute{\Psi}_\tau\rangle$.

b. Second layer is built up of isometries which form (the state $|\Psi_{\tau+1}\rangle$ of $N_\tau /2$ wires) single wires (the state $|\acute{\Psi}_\tau\rangle$ of N_τ wires) pairs of nearest neighbor wires where $N_{\tau} = 2^{\Theta - \tau}$.

This means that the so-called entanglement renormalization, or what is also known as the real space coarse graining, where a certain class of it is implemented by $\mathcal{M}$ [68]. To create a series of coarse-grained lattices $\{\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2, \dots\}$ and effective Hamiltonians $\{H0, H1, H2, \dots\}$, it is necessary to transform a lattice's sites $\mathcal{L}_0 \equiv \mathcal{L}$ and operators defined on $\mathcal{L}$ like Hamiltonian H_0, certainly using the MERA. the operator contained into a causal cone $C^{[s]}$ is mapped. when mapped into $\mathcal{L}_\tau, 1 \leq \tau \leq \log_2(N) - 1$, the local operators into $\mathcal{L}$ remain local as defined on site $s \in \mathcal{L}$. With $O(2^\tau)$ sites in $\mathcal{L}$ of a D-dimensional hypercube one site in $\mathcal{L}_\tau$ is obtained. Since the foregoing, at length scales greater than $2^{\tau/D}$, $|\Psi_\tau\rangle$ can be comprehended as preserving $|\Psi\rangle$'s structure. In a $1D$ lattice a MERA can encode accurate approximations to a quantum critical system's ground state in a far more efficient method than an MPS [68] , χ_{MPS} terms in the Schmidt decomposition provide for correlations between the right and left chain's halves in an *MPS*. In the Schmidt decomposition, MERA may account for around $2\chi \log_2 N$ terms. As a result, the state saved with MERA of χ^{-level} wires and requires an *MPS* with $\chi_{MPS} = 2\chi \log_2 N$ [65].

For finding entanglement entropy $S(\rho^{[B_l]}\ )$ of the block B_l of l, as well as the distance r between 2 sites s_1 and s_2, a power-law (illustrated by $log\ l$) shows the correlators $C_2(s_1, s_2)$ decay with the distance r between those sites, as well as adjacent sites scales [69]–[71]. After $O(log\ r)$ transformations shown in left side of Figure 3.29 (the λ-shaped causal cone $C^{[s_1,s_2]}$), $\rho^{[s_1,s_2]}$ is obtained after taking into account a sequence of density matrices $\{\rho_{\acute{\tau}}, \ldots,\ \rho_2,\ \rho_1, \rho^{[s_1,s_2]}\}$ considering $\overline{\tau}\ \approx\ D\ \log_2 r$ in the density matrix $\rho_{\overline{\tau}}$ corresponding to a hypercube of $\mathcal{L}_{\overline{\tau}}$'s 4$D$ sites for the purpose of calculating $\rho^{[s_1,s_2]}$. A power-law scaling for $C_2(s_1, s_2)$ can be obtained with a constant factor $z\ <\ 1$ causing these transformations reduces correlation.

$$C_2(s_1, s_2) \approx z^{\log r} = r^{-q}, \qquad q = \log\frac{1}{z} \tag{3.18}$$

Equation 3.18: Power-law scaling [66].

Through a path of length $O(\log r)$ in $\mathcal{M}$, at a distance r there two sites are connected in $\mathcal{L}$ to describe states which makes MERA ideal for quasi-long-range order. The causal cone of B_l considering the entropy $S\big(\rho^{[B_l]}\ \big)$ and shrinks exponentially fast with τ for the density matrix $\rho^{[B_l]}$ of a hypercube B_l made of l-D sites [65]. While the right side of Figure 3.29 computes $\rho^{[B_l]}$ of $\acute{\tau}\ \approx\ D\ \log_2 l$ from the density matrix $\eta_{\overline{\tau}}$ and through a sequence of density matrices $\{\eta_{\overline{\tau}}, \cdot \cdot, \eta_1, \rho^{[B_l]}\}$ for a hypercube made of (at most) 4D sites of lattice $\mathcal{L}_{\overline{\tau}}$. Applying a layer of isometries to get Density matrix η_τ from $\eta_{\tau+1}$, as well as applying a disentangler layer which have no effect on entropy of $\eta_{\tau+1}$ as well as tracing out $n_\tau =\ O\big(2^{(\overline{\tau}-\tau)(D-1)\ /D)}\big)$ boundary sites, following one of them increases the entropy by at most $\log_2(\chi)$ bits, which leads to an increase in total at most $n_\tau \log_2(\chi)$ bits after an increase in entropy ΔS_τ, and this in turn achieves the entropy of $\rho^{[B_l]}$ [65].

$$S\big(\rho^{(B_l)}\big) - S(\eta_{\overline{\tau}}) = \sum_{\tau=1}^{\overline{\tau}} \Delta S_\tau \leq \log_2(\chi) \sum_{\tau=1}^{\overline{\tau}} n_\tau \tag{3.19}$$

Equation 3.19: The entropy achieving, at most $\mathbf{4^{D}\ \log_2(\chi)}$ acts $\boldsymbol{S(\eta_{\overline{\tau}})}$ [65].

The number of dimensions must be checked for MERA, first case when D = 1, the constant sets the traced number of sites n_τ for any $\tau, 1\ \leq\ \tau\ \leq\ \overline{\tau}\ =\ O(log\ l)$.

$$S\left(\rho_{\{l\}}\right) - S(\eta_{\bar{\tau}}) \leq \log_2(\chi)\, c\bar{\tau} = O(\log l) \tag{3.20}$$

Equation 3.20: The number of dimensions utilizing MERA in first case [66].

The second case, when $D > 1$, and by the contribution from small τ results in the upper bound for the $S\left(\rho^{(B_l)}\right)$ being dominated, and as a result the n_τ decays exponentially with τ.

$$\begin{aligned} S\left(\rho^{[B_l]}\right) - S(\eta_{\bar{\tau}}) &\leq log_2(\chi) \sum_{\tau=1}^{\bar{\tau}} 2^{(\bar{\tau}-\tau)\frac{D-1}{D}} \approx \log_2(\chi)\, 2^{\bar{\tau}\frac{D-1}{D}} \\ &\approx \log_2(\chi)\, 2^{D\log_2(l)\frac{D-1}{D}} = \log_2(\chi)\, l^{D-1} \end{aligned} \tag{3.21}$$

Equation 3.21: The number of dimensions utilizing MERA in second case**,** this means that according to a boundary law $\boldsymbol{S^{[B_l]}} \approx \boldsymbol{l^{D-1}}$ that block entanglement at $\boldsymbol{D} > \mathbf{1}$ can be supported by MERA [65].

G. Vidal has demonstrated in his paper that Computational benefits can be achieved by tailoring the MERA to the details of a task where throughout $\mathcal{M}$, the amount of levels χ might change especially since The MERA can be used to assimilate the symmetries of $|\Psi\rangle$ where a specific MERA can be constructed in dimensions $D = 2$ as an example to depict triangular lattice states, or to account for the boundary conditions' varieties, or of a lattice with bulk defects, linear or random vacancies [65].

The MERA is only dependent on roughly $\chi^4 \log_2(N)$ parameters if in case a layer of $\mathcal{M}$ that all the tensors in are selected to be identical, so, with periodic boundary conditions in the system, cyclic shifts by one lattice site occur, when $|\Psi\rangle$ is invariant during translations, more significant improvements are gained. More precisely, because of the internal symmetry, the tensor in $\mathcal{M}$ has a sequence of restrictions that are based on fewer parameters. However, even in an infinite lattice, The states that are not affected by entanglement renormalization transformations are the ones that save the most substantial [65], [68]. In this case, all tensors into $\mathcal{M}$ are identical, and MERA is only dependent over $O(\chi^4)$ parameters. A MERA's potential is not limited to the portrayal of individual states but it has the ability to produce an endless number of entangled states $\left|\Psi_{\{\phi_r\}}\right\rangle$, by sending the state $\otimes_r^N = 1 \left|\phi_r^{[r]}\right\rangle$ to the incoming wires r of quantum circuit C instead of $\otimes_r^N = 1\left|0^{[r]}\right\rangle$, all of them are represented by a MERA with just the isometric tensors differing. that fulfill.

$$\left\langle \Psi_{\{\phi_r\}} \middle| \Psi_{\{\acute{\phi}_r\}} \right\rangle = \prod_r \left\langle \phi_r^{[r]} \middle| \acute{\phi}_r^{[r]} \right\rangle \tag{3.22}$$

Equation 3.22: Encode quasi-particle excitations in only one (generalized) MERA [66].

3.8 DECODING FOR TOPOLOGICAL CODES WITH NEURAL-NETWORKS

One of the most important computational problems faced by generic stabilizer codes is Finding optimal correction of errors regardless of the complexity of noise models and developing effective and efficient decoders is still a challenging task despite using some structure to simplify this encoder. Feedforward neural network decoders are a type of decoder that can be adopted in the methodology of this thesis. Because neural networks do not need previous knowledge of the underlying noise model, they simplify the process of constructing high-performing decoders [72]. Logical information is encoded into quantum error-correcting codes to be protected [73]. Without changing the encoded information, stabilizer operators be measured utilizing Stabilizer codes, making them one of the most effective quantum code types. Then it implements a recovery operation for the purpose of correct errors [74], followed by an application of a decoder algorithm, which is a classical algorithm to deal with the available classical data for finding an appropriate correction. Computationally, the noises generated in the models make it difficult to obtain optimal decoding of generic stabilizer codes [75]. For example, stabilizer generators are geometrically local for topological stabilizer codes where they return -1 indicating errors' presence on some qubits into their neighborhood when they get an unsatisfied stabilizer as an outcome. Accordingly, If the encryption has any structure, decoding becomes more tractable [76]–[80]. Many decoding techniques have been developed, including tensor networks [81], [82], Minimum-Weight Perfect Matching algorithm [83]–[85], and cellular automata [86]–[88]. Decoding strategies in Taylor-made is done in two-step:

i. step 1: finding a Pauli operator for unsatisfied stabilizers, this function returns corruptly encoded data to code space. In the end, all stabilizers do but the attempted Pauli correction, which could have resulted in a non-trivial logical operator when combined with the initial error.

ii. Step 2: A non-trivial logical operator would probably have been introduced upon the first step to be determined by a feedforward neural network for the purpose of account the recovery. Where this step is a classification problem that fits machine learning well.

In the future, neural-network decoding might be applied in a variety of small-scale quantum devices being a promising error-correction method because due to its adaptability, especially if the most common errors' sources aren't adequately defined. where the training costs and neural network parameters scale as the coding distance increases.

M. Nishad et al. in [72] introduce a different concept of an effective error rate that makes simple comparing various noise models into threshold error rates. Also, they argue about how neural decoding works in different noise models, like linked de-polarizing noise. The approach of training deep neural networks is anticipated to have a substantial influence, as it may result in faster convergence and enhanced neural network ultimate performance in quantum error correction applications.

3.8.1 Topological Codes For Error Correction

3.8.1.1 Topological Stabilizer Codes

A stabilizer group S determines quantum error-correcting codes, due to the relevance of stabilizer codes [73]. When where $P_i \in \{I, X, Y, Z\}$ where $-I \notin S$, the n-qubit Pauli operators $P_1 \otimes \ldots\ P_n$ are used to construct the Pauli group one of whose subgroups is the stabilizer group S. All the elements of S are encoded as the (+1)-eigenspace in the code space. Logical Pauli operators $\overline{L} \in \mathcal{L}$ are put to use in the Pauli grouping of normalizer S components that are unique to the S-stabilizer. Pauli $\overline{L} \neq \overline{I}$ is a non-trivial logical operator implemented by the operator L to be the product of Pauli operators, where the product does not belong to the S-group and instead commutes with the stabilizer group. It's important to keep in mind that the distance between two codes is using non-trivial logical weight of minimal-support Operator Pauli. The stabilizer code's physical qubits may be affected by noise if the latter moves the region of the code space containing the encoded logical information to the perimeter of the code. Because the corresponding original encoded state cannot be detected by monitoring stabilizer generators, the projection of system flaws onto certain Pauli operators provides some insight into the nature of those flaws. -1 measurement outcome is returned by a collection of unsatisfied stabilizers which called a syndrome, A decoding algorithm utilizes syndrome like classical input. A recovery Pauli operator may be found using a decoding algorithm, this adds to the corrupted encoded state being returned to code space. In this analysis, we focus on a subset of stabilizer codes known as CSS codes used in the final product of either Z-type syndrome Pauli operators or X-type syndrome Pauli operators in an appropriate manner for correcting X-type errors and Z-type errors independently [89]. As a distinguishing property, topological stabilizer codes' generators demonstrate extremely strong noise resistance [76]–[80]. By putting physical qubits into a lattice in parallel, each stabilizer generator may be kept at a fixed number of qubits within a geometrically limited local area, Since it is impossible to apply the unitary operating logic of the Pauli operator to local physical qubits, Consequently, there is no longer any need Modify the level of measurement complexity needed for local stabilizers in order to attempt to increase error-correction capabilities and distance of topological code during system expansion. An example about a topological stabilizer code is as follows:

a. A triangular toric code.
b. A triangular color code.

In the first example, Figure 3.30 (a) illustrates on a two-dimensional lattice with a border, The triangle color code has been established, its facets $f \in F$ are 3-colorable and whose vertices are 3-valent. Vertices are used to identify qubits. The color code represents a CSS code, where a CSS code stabilizer group can be represented as follows:

$$S_{CC} = \langle X_f, Z_f \backslash f \in F \rangle \tag{3.25}$$

Equation 3.23: CSS code stabilizer group [73].

To utilizing the Z-type and X-type syndrome, independently Z-type and X-type errors can be improved due to support for X_f and Z_f are Z operators and Pauli X over all qubits belonging to face $f \in F$ [72].

(a) A hexagonal lattice patch with 3-colorable faces and 3-valent vertices has a 2*D* triangular color code. Both *X*- and *Z*- stabilizers have been supported by every face. a logical *X* operator has been implemented by the string of Pauli *X*- operators (orange⊗) as same as a logical *Z*- operator has been implemented by the string of Pauli *Z*- operators (yellow⊕), both of them connect all three boundaries.

(b) 2*D* triangular toric code with a twist. *X*- and *Z*- stabilizers are supported by dark and white faces, respectively. Stabilizers Pauli *X* and Pauli *Z* are mixed products based on the coloring of mixed dark/white faces along a 1*D* defect line (dashed line). Two comparable representations of a logical *Z* operator are shown by red and blue strings. The string transforms from *X*-type (blue⊗) to *Z*-type (blue⊕) as it crosses the defect line.

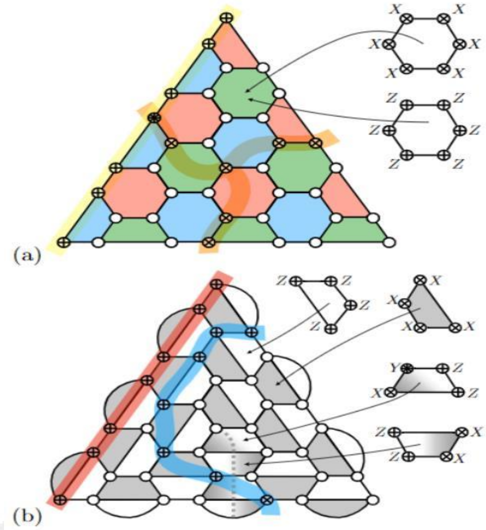


Figure 3.30: 2D triangular color and toric code [72].

In the second example, with the equivalent physical qubit arrangement of a triangle color code, the triangular toric code may be defined with a twist. [90]. Figure 3.30 (b) illustrates its lattice may be produced by adding extra edges, preserving all the vertices, as well as changing certain faces from color-code lattice. With exception of faces alongside 1*D* defect line which called "mixed", a resultant lattice is faces are 2-colorable and 4-valent. *Z*-type Z_g stabilizers and *X*-type X_f are supported with white $g \in F_W$ and dark $f \in F_D$ faces indicating the stabilizer generator type associated to that face. The products of *X*-Pauli operators and *Z*-Pauli operators can be mixed based on coloring of mixed faces $h \in F_M$ so that S_h stabilizers are defined. Whereas, to move with X_f and Z_g for all $f \in F_D$, $g \in F_W$, $h \in F_M$, it is necessary for the stabilizers to be achieved along the defect line, the choice of mixed stabilizer generators. The full stabilizer group can be achieved with the Equation 3.24:

$$S_{TC} = \langle X_f, Z_g, S_h \backslash f \in F_D, g \in F_W, h \in F_M \rangle \qquad (3.26)$$

Equation 3.24: The full stabilizer group achieving [73].

Due to the use of combined stabilizer generators, it is not possible to perform independent decoding of X and Z faults. Deformable non-contractible 1D string-like operators may be thought of as the logical Pauli operators in 2D topological stabilizer codes. Figure 3.30 shows how the logical operators are delimited by boundaries in both toric codes and triangular color.

3.8.1.2 Quasiparticle Excitations

The commuting Hamiltonians explain the construction of a correlation between quantum error-correcting codes and quantum many-body systems. A commuting stabilizer Hamiltonian *H*(*S*) is determined by group *S*. Both the stabilizer Hamiltonian of the color code and the stabilizer Hamiltonian of the toric code are selected with a twist: the sum of the stabilizer generators of S is given a negative sign as follows:

$$H_{CC} = -\sum_{f\in F} X_f - \sum_{f\in F} Z_f \tag{3.27}$$

Equation 3.25: The color code' stabilizer Hamiltonian with a twist [73].

$$H_{TC} = -\sum_{f\in F_D} X_f - \sum_{g\in F_W} Z_g - \sum_{h\in F_M} S_h \tag{3.28}$$

Equation 3.26: The toric code' stabilizer Hamiltonian with a twist [73].

Due to mutual commuting for all terms in the Hamiltonian H of the stabilizer (S), every eigenstate of a term must also be an eigenstate of H(S), (1)-eigenspace is the code space specified for each he ground space of *H*(*S*) that coincides with the elements of *S* considering that stabilizer generator eigenstates can only have ±1eigen value. A localized quasiparticle excitation is one consequence of the error information contained when taking into account any anticommuting Pauli fault with specific stabilizer generators in the related quantum many-body system. The encoded logical state (ground state) is moved outside code space (the ground space) by the error. The number of violated stabilizers would lead to an excited state whose energy is proportional to the number of stabilizers that were initially in place when it is greater than the ground space energy. Hamiltonian expressions (quasiparticle excitations identify as the unsatisfied stabilizer terms) [76], [91]–[93].

(a) In the bulk of the color code: The three unsatisfied *Z*- stabilizers on neighboring faces are magnetic excitations (red, green, and blue⊠1) are created by a single *X*- error (white⊗1), while a pair of magnetic excitations (red⊠2) are created by a string of *X*-errors (white⊗2), finally a single electric excitation (blue⊠3) is created by a string of *Z*-errors (white⊕3).

(b) In the bulk of the toric code: the twist two unsatisfied *X*-stabilizers on neighboring dark faces are a pair of electric excitations (gray⊠1) are created by a single *Z*-error(white⊕1), while a single magnetic excitation (white⊠2) on the rough boundary creates is created by a single *X*-error (white⊗2), finally a string of errors(white⊗3and⊗3) can create a pair of electric (gray⊠3) and magnetic (white⊠3) across the defect line (dashed line).

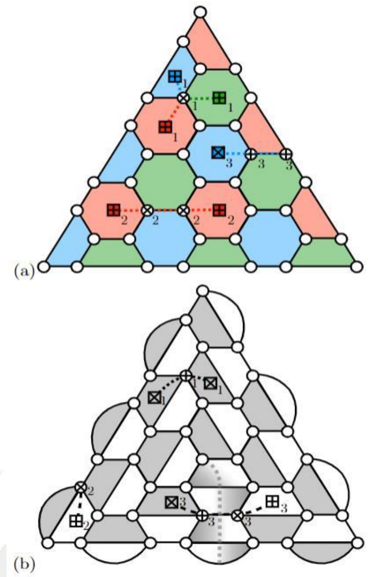


Figure 3.31: Excitations of quasiparticles in 2D triangular color codes and triangular toric codes [72].

Assuming that the excitation is supported by the color of the face which will be indicated by the symbol K does not represent the excitation electric, m_K denotes the excitation magnetic field, which is determined by whether the unmet stabilizer is of the *Z*-type or *X*-type. On the other hand, faces of any color are supported by the color code excitations i.e., $m_K$and e_K for any $K \in \{R, G, B\}$ because there are only e_D and m_W in the toric code. Local Pauli operators can create excitations geometrically, so it is necessary to understand the configurations of these excitations (fusion rules for topological stabilizer codes) that result from any Pauli errors and how to combine them.

Figure 3.31(b) shows creating two excitations (magnetic or electric) of the same type in the toric code on the qubit, a Pauli Z-error or Paul X-error on a single qubit violates two *X*-type stabilizers or two *Z*-type stabilizers in the bulk of the system on faces absolutely. More precisely, there is no excitation at the location where independently created the same excitation on a face $f \in F$ by two errors with non-overlapping support. It's worth noting that 1 means there's no excitement in the toric code fusion rules is detailed in the excitation's equation of the same kind in the bulk only being created using geometrically local operators or destroyed in pairs, according to Equation 3.27:

$$e_D \times e_D = m_W \times m_W = 1 \tag{3.29}$$

Equation 3.27: The excitation's equation of the same kind using geometrically local operators or destroyed in pairs [73].

Figure 3.31 (a) shows the fusion rules for the color code. By the way, the fusion rules' equation of the color code is a complex comparing with its counterpart as shown by the Equation 3.28 and Equation 3.29:

$$e_K \times e_K = m_K \times m_K = 1\,, K \in \{R, G, B\} \tag{3.30}$$

Equation 3.28: The fusion rules' equation of the color code part 1 [73].

$$e_R \times e_G \times e_B = m_R \times m_G \times m_B = 1 \tag{3.31}$$

Equation 3.29: The fusion rules' equation of the color code part 2 [73].

Figure 3.31 (a) clearly demonstrates that the color code does not elicit any excitement, unlike the case When using a toric code, this occurs when two effects that are identical in both type and color are combined. Triple excitations may be created (through the local operator) or destroyed in a manner analogous to the toric code when the Pauli Z-error or Pauli X-error of a single qubit is taken into account. Once three adjacent faces (X- or Z-type stabilizers) have been worn down by an X-error, a magnetics' triple or an electric excitations' triple is generated. It is possible to create or destroy a single electric or magnetic excitation after determining the topological stabilizer codes to be considered on bordered lattices by operating with on the qubits at the boundary of the system a local Pauli operator, where the type of permissible excitation is controlled by the type of border [94]. Only at the limits of the triangular toric code [77] can a single electric excitation or single magnetic excitation be created, as shown in Figure 3.31 (b). In contrast to what is shown in Figure 3.31 (a), a single electric excitation may result in the production of magnetic excitations of the corresponding color [78]. The existence of red, green, or blue borders in the triangle color code is indicative of this. According to the fusion rules, a quasiparticle excitation cannot change type until it passes through a defect line (a transparent do main wall) produced by proper adjustment along the stabilizer generators, and this movement along a route is defined as an excitation changing its location after forming excitations' pairs in that direction and fusing them with the starting path

[95]. Finally, as shown in Figure 3.30 (b), for the mixed products of Pauli X and Z operators at a twist point in the triangular toric code, stabilizers on faces must be chosen that are in intersection with the defect line, which is the crossing boundary at which it turns (electric\magnetic excitation $e_D \leftrightarrow m_W$) and vice versa. Figure 3.30 shows a single excitation can be created by executing an appropriate 1D string-like logical Pauli operators over one of the boundaries, where it is relocated into the 2D topological stabilizer code's bulk to the other boundary where it can annihilate [96].

3.8.1.3 Topological Code Decoding As A Classification Issue

It has been noted that some unsatisfied stabilizers are configured when encoded information is moved outside of the code space due to general errors. For stabilizer codes, all installers are restored after a suitable recovery is found by the classical algorithm when its input is Syndrome with a yield +1 measurement outcome as the decoder whose success depends on recovery combined with error given that a Pauli operator is the recovery operator in the state of not performing any non-trivial logical operator. Decoding may be viewed of as a procedure that removes quasi-particle excitations from the system before returning it to the stabilizer Hamiltonian's ground space. In order to clarify the work of the decoding of topological codes, the following questions must be answered: What happens at the topological stabilizer code's lattice? How can the excitations move in order to remove them?

Figure 3.32 represents an $excitation\ graph\ G = (V, E)$ as an attempt to answer the above questions, which owns the vertices V for the quasiparticle excitations, or at least for the possible places thereof, where each electric as well as magnetic excitation has its own vertex.

Assuming In two types, the color and toric codes, there is local Pauli operator P_{v_1,v_2} for two different vertices v_1, v_2 where an edges $\{v_1, v_2\}$ is identified in E: $\{v_1, v_2\} \in E$ is working on ($v_1, v_2 \in V \setminus$ the special boundary vertex $\{w\}$), over one or two nearby qubits, local operators are supported respectively, then an excitation can be moved from v_1 to v_2 by geometrically local support without creating any other excitations. If a single excitation at v can be formed locally, then an edge $\{v, w\}$ can connect the boundary vertex w and $v \in V \setminus \{w\}$.

A single-qubit operator cannot move a single excitation in the bulk of the system without causing other excitations, but an excitation may be moved between two nearby faces f and g of the same color by a single-qubit Pauli *X*-error or *Z*-error. Thus, between v_f and v_g, an edge v_f, v_g can be added to the collection of edges E of G. the single- qubit or two-qubit operator connects vertex v_f with the boundary vertex w to generate a single excitation on f.

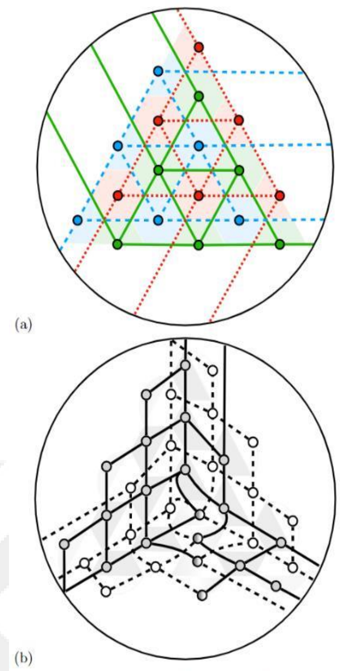


Figure 3.32: An excitation graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ for both toric and color with twist code. The topological code lattice's face $\boldsymbol{f}$ with the vertices set $\boldsymbol{V}$ of $\boldsymbol{G}$, and the boundary vertex $\boldsymbol{w}$ into $\boldsymbol{V}$ [72].

The quasiparticle excitations from the toric and color codes can be eliminated efficiently using the excitation removal algorithm, which can be simplified as follows:

Table 3.3: Using the excitation removal algorithm to convert the decoding problem to a classification problem [73].

Algorithm 1: excitation removal
Require: the excitation graph $G = (V, E)$
Input: positions $U \subset V \backslash \{w\}$ of excitations
Output: Pauli operator R_U removing all excitations
initialize $R_U \leftarrow I$
for every $u \in U$:
1. find the shortest path $(v_1, v_2, \ldots, v_n)$ in G between $u = v_1$ and the boundary vertex $w = v_n$
2. find an operator $P_u = P_{v_1,v_2}, \ldots, P_{v_{n-1},v_n}$ corresponding to the path $(v_1, v_2, \ldots, v_n)$
3. $R_U \leftarrow R_U P_u$
return R_U

The algorithm takes U as input, but not Q which represents some Pauli error operator to create the excitation $U \subset V\backslash\{w\}$ in the system considering there is a path $(v_1, v_2, \ldots, v_n)$ between $u = v_1$ and the boundary vertex $w = v_n$ within the excitation graph G, where there is excitation $u \in U$ where $v_i \in V$ and the edges$\{v_i, v_{i+1}\} \in E$. By which along the given path the local Pauli operators $P_{v_i,v_{i+1}}$ is defined which in turn produces an operator $P_u = \prod_{i=1}^{n-1} P_{v_i,v_{i+1}}$ which transfers the excitation from u into it's the annihilate in a boundary. The state has been returned to the ground space by an operator $R_U = \prod_{u \in U} P_u$ which represents the output of the algorithm after all the excitations have been removed, thus $R_U Q \in \mathcal{L}$. It is likely that a decoder algorithm would perform rather poorly in the case of the implementation of some non-trivial logical operator with the initial error Q by the output R_U combined. The task of the excitation removal algorithm is to transform the excitation problem from decoding to classification, by assigning a small set of labels to a high-dimensional dataset's elements. Where positions $U \subset V\backslash\{w\}$ of the excitations are defined in order to implement the trivial logical operator (Pauli operator Q) after removing all the excitations (the excitation configuration U generated by Q) ultimately producing the operator R_U.

A recovery operator $R_U L$ choosing to get a logical operator $\overline{L} \in \mathcal{L}$ had implemented by operator L to act $R_U Q$. The most likely equivalence errors' class creating U can be found after efficiently decode, $R_U Q$ splits error operators Q to classes of equivalence designated by distinct logical operators $\overline{L}$ provided using $R_U Q$. The method achievement is to use the logical operator $\overline{L}$ to label U, where $\overline{L}$ is implemented by the excitation removal algorithm's output R_U as well as any operator Q from the most probable errors' class. Machine learning techniques, particularly artificial neural networks, are well-suited to such a challenge.

3.8.2 Performance Of Neural-Network Decoding

3.8.2.1 Neural Decoders

Decoding is successful if it solves the following problem:

$R_U Q$ executes the logical operator $\overline{L}$ labeled from the set of logical operators $\mathcal{L}$ and assigned by some unknown Pauli operator Q for any configuration of excitations $U \subset V\backslash\{w\}$ for algorithm input U and output R_U. After the excitation removal algorithm transforms the excitation problem from decoding to classification, using the feedforward neural networks technology becomes an appropriate approach to dealing with the classification problem. The input layer and output layer make up neural networks, with the hidden layer in between. As shown in, $H_d + 2$ layers of a neural network will be training for the distance code d, with suppose the configuration of excitations U as input layer encodes and, each of hidden layers H_d consist of N_d nodes fully connected with nodes ν from next hidden layer which evaluates an activation function $\sigma(w_\nu \cdot o_l + b_\nu)$ on the output o_l of nodes from the previous layer as well as associate the weights w_ν and the biases b_ν where $\sigma(x) = max(0, x)$ has been selected as the rectified linear unit activation function. And finally, The SoftMax classifier is used in the output layer to convert an output vector to a discrete probability distribution in order to characterize the chance of distinct logical operators $\overline{L} \in \mathcal{L}$ being implemented using $R_U Q$.

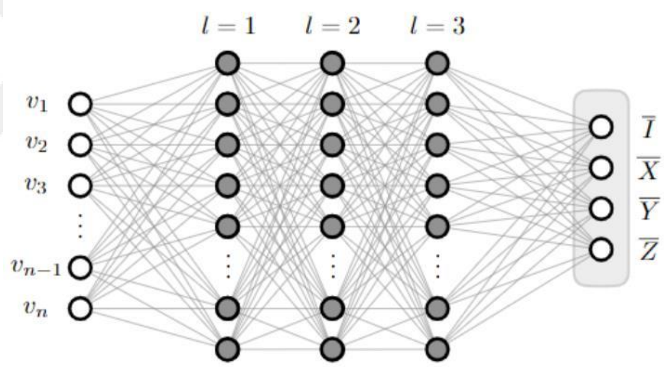


Figure 3.33: The input layer where all of the initial excitation configuration $\boldsymbol{U} \subset \boldsymbol{V}\backslash\{\boldsymbol{w}\}$ is encoded and the output where the probability of each logical operator $\overline{\boldsymbol{L}} \in \{\overline{\boldsymbol{I}}, \overline{\boldsymbol{X}}, \overline{\boldsymbol{Y}}, \overline{\boldsymbol{Z}}\}$ assigned to the input configuration $\boldsymbol{U}$, also $\boldsymbol{H_d} = \boldsymbol{3}$ hidden layers equal number of nodes $\boldsymbol{N_d}$ in a feedforward neural network where each hidden layer node is fully connected with all subsequent hidden layer nodes [72].

Some unknown operators Q create a configuration of excitations $U \subset V\backslash\{w\}$ that we aim to retrieve or a recovery operator R using a neural decoder algorithm given that some a priori unknown noise models determine the selection error operators Q.

The mechanism of action of the neural decoder can be summarized in two steps:

i. The excitation removal algorithm is used to remove quasiparticle excitations that result in R_U recovery operator.

ii. guess the most likely errors Q resulting from:

a. U. or

b. logical operator $\overline{L}$ which will be executed later using $R_U Q$. This is using a neural network where the output will be operator $R_U L$ embedded in R where the logical operator $\overline{L}$ implements L.

Table 3.4: Neural decoder algorithm [73].

Algorithm 2: neural decoder
Require: excitation removal algorithm, trained neural network
Input: locations of excitations $U \subset V \backslash \{w\}$ created by some unknown operator Q
Output: recovery operator R
using the excitation removal algorithm with U as the input, find an operator R_U
using the neural network with U as the input, find the logical operator $\overline{L}$ implemented by $R_U Q$
$R \leftarrow R_U L$, where L is any operator implementing $\overline{L}$
return R

The attributes of the neural decoder

i. The neural network successfully recognizes the $R_U Q$ which implements logical operator L.

ii. The neural decoder reduces to matrix multiplication, so the output of the trained neural network was determined efficiently.

iii. For training the neural network in *step*2, no knowledge of topological coding is required because information about fusion rules and topological code lattice is included in the excitation graph in *step*1.

Supervised learning is used to train a neural network on a dataset of pre-classified samples before use. The noise model defines the probability distribution for the Monte Carlo sampling used to generate the Sample Pauli errors. The excitation configuration $U \subset V \backslash \{w\}$ is input to neural network wherein corresponding syndrome is specified for each generated error configuration Q, followed by finding the Pauli operator R_U using the excitation removal algorithm in order to specify the logical operator $\overline{L}$ to be executed by $R_U Q$ causing the output to have assigned a logical operator $\overline{L}$ to each input excitation configuration U from logical operators' set $\mathcal{L}$.

4. THE EXPERIMENTS

4.1 INTRODUCTION

High-performance quantum circuit simulators are made possible by the open-source library of rapid prototyping included in the TensorFlow Quantum (TFQ) supplied by the TensorFlow framework. This library may be used with either classical or quantum data inside hybrid quantum models. The researcher aspires to provide an overview of the building blocks and the software architecture by simplifying the hybrid quantum-classical neural networks' theory via supervised learning utilized in the Convolutional Neural Network Hybrid Quantum-Classical Classifier as a BASIC QUANTUM APPLICATIONS to replicate and expand the previous quantum machine learning achievements in the second-generation using the various features of the TFQ due to in contrast to the first generation, Noisy Intermediate-Scale Quantum (NISQ) processors are emergence more widely available [20], which depended on increasing the area of the state exponentially with the number of qubits while performing fast linear algebra on it [20], [21], [97], in addition, the researcher shows how to use the TFQ to solve sophisticated quantum learning problems including reinforcement learning, generative adversarial networks, classification of quantum phase transitions, varying quantum eigen-solvers, sampling thermal states, Hamiltonian learning, layer wise learning and meta-learning. Information hidden within a data set can be extracted by means of algorithms and statistical models that are built with machine learning (ML). The same basic probability distribution of unseen data is utilized to generate predictions through learning a model from a dataset, and because of the abundance of data and computational resources provided by the inference method, it is dominant over other methods [98]. The suitability of the model to the data set is determined by reducing the function of the outputs of the model, which is set by adjusting the parameters of the layers constituting the networks representing the data set under the term loss function within the deep learning method [99]. It is difficult for classical devices to simulate quantum systems because of the entanglement and superposition that occurs in these systems, which are described by generalizing probability theory and this is the essence of the working principle of quantum computing [100].

First, the data must be transformed into a quantum state in order to apply appropriate machine learning algorithms to classical data [101], unlike the quantum machine learning's first-generation which depended on increasing the area of the state exponentially with the number of qubits while performing fast linear algebra on it. In contrast to the first-generation, the second-generation (a research area of this thesis) is Quantum Machine Learning utilizes its new algorithms called parameterized quantum circuits (PQCs) or Quantum Convolutional Neural Networks (QCNNs) [23], [102] similar to classical deep learning algorithms. In order to optimize the parameters of a QNN in connection to a cost function, these novel algorithms learn the representation of the training data and use either gradient-based approaches [36] or black-box optimization heuristics [103]. Quantum computing is enhancing essential indicators of learning performance, such as representation capacity, training efficiency, and Generalization power, which NISQ processors will require to function in concert with conventional coprocessors that will be integrated with co-parameter quantum circuits. Among the ways to utilize outer-loop optimizers for QNNs to combine quantum and classical computations are classical computers. The comprehensive algorithm through training QNN is so-called Variational Quantum-Classical Algorithm whose proposed architecture includes Quantum Convolutional Neural Networks (QCNN) [12] which is a work principle on this thesis. It should be noted that when a network architecture is randomly initialized or lacks any structure, it creates an obstacle to performance of scaling-based and gradient-free optimizations known as barren plateaus [104]. The development of QML techniques requires a working knowledge of both quantum computing and machine learning. Permeates our work, taking advantage of the characteristics offered by Quantum computing packages that use the elements of the highest quantum computing libraries in machine learning in order to create basic gates within simulators, measure results and take advantage of quality control tools, similar to commercial and research data analysis software, where this package is handled with the aim of discovering specific a quantum data source's specific properties like a complex simulation of a device or a quantum sensor by putting a proposed a convolutional neural network's quantum equivalent into practice due it already translationally invariant. The experiments the researcher conducted in the section Experiments were well-suited to the use of these libraries.

At the beginning of 2020, Google announced a new computing platform called TensorFlow Quantum (TFQ) very similar to the original TensorFlow for quantum machine learning as well as made available to public under Python support and containing hybrid quantum-classical ML models for rapid prototyping [105]. Google has provided guidance on Design, testing, and development mechanisms of learning models working alongside the symbolic mathematics "Sympy" library [106] and the quantum logic circuit design "Cirq" library [35].

In 2018, the Google AI Quantum Team utilized a software stack that enables quantum computing to build and develop "Cirq" to work over TFQ as be design package that provides several capabilities but is a couple of years older. Our TFQ model includes encoding and convolutional circuits based on the amplitude of the quantum signal. In 2006, a completely functional Characteristics of Computer Algebra System (CAS) with many components including Calculus, Combinatorics, Physics, Geometry, and mathematics was utilized to build the "Sympy" library [106]. For deep learning computation models, "Sympy" is one of the two components (the TensorFlow graph and placeholders) that the TensorFlow library relies on for the purpose of using control parameters within the quantum model, where the parametric quantum variable placeholders are considered inside the model graph wherever quantum calculation provides intermediate values.

As a portion of this thesis, the researcher utilized Python version 3.7 for experimentation, coding, and development. Simple packages are managed and TFQ environment is controlled by creating separate environments utilizing Anaconda with 'conda-forge' counterpart to install most packages. If a particular package cannot be installed via these two channels, the Package Installer for Python (PIP) Instead, when developing code, "Jupyter" Notebook and Visual Studio Code as the Integrated Development Environments were used.

"Numpy" library was used in applying many post-analysis steps as well as pre-processing steps as it is used in vector/matrix operations and linear algebra, where it was used for the purpose of data pre-processing for predictive analytics as well as in learning methods along with "Matplotlib" library [107].

Extensive plotting capabilities are leveraged for visualization using a multilanguage library "Matplotlib.pyplot", due to its close association with the "Numpy" library when used in most graphs in terms of practical use and internal development [108] and for including functions for graphs such as pie charts, density, histogram, line, scatter among others. Visualization has been used as the main means of communicating the thesis preparation methods and results by facilitating the understanding of many modifications. Using Microsoft Word, this thesis was written using the collecting and arrangement of sources and references from the online Mendeley, and unique graphics from plots of data were created using the online Microsoft Visio program as well as Microsoft excel.

4.2 BACKGROUND

exploring quantum machine learning applications for QNNs demands the researcher adopted the supervised classification technique [33] in order to isolate the information containing the label using a form of compression using hierarchical models [109]. Before extracting information from the non-local subspace by performing some non-entangled quantum transformation, a quantum system subspace, or a non-local subsystem in quantum data case can be used to include the classical hidden parameter (classification case discrete label or real scalar in the case of regression). choosing an architecture for a neural network model utilizing Training data wherein learning is a common filter set that are applied uniformly across all data segments after linking translationally invariant neural networks parameters across space. Cluster states can be used as the initial states to quantum computation measurement-based as types of quantum data with translational symmetry [50]. A range of successfully and wrongly produced cluster states will be adopted to create our training data, each labeled as a type of supervised class detecting errors in their preparation. wherein the phases classification, the degree of entanglement is high near quantum critical points. Because basic cluster states are translationally invariant, quantum neural networks can be included as an extension of the convolutional neural networks' spatial parameter-tying. The MERA (Multiscale Entanglement Renormalization Ansatz) network [65] is a highly entangled wavefunctions' hierarchical representation, as it has been adopted by Cong, et al. [12] In constructing the Quantum Convolutional Neural Network (QCNN) structure as it is concerned with condensed matter physics and as in deep neural networks the wavefunction's entanglement getting renormalized (coarse-grained) is a wavefunction's compressed representation synchronously [110]. A Hybrid Quantum Convolutional Neural Network (HQCNN) would be produced by Including more processing including classical neural network postprocessing for extending the QCNN architecture. More precisely, on one tier of the QCNN hierarchy architecture, partly disentangling the input state to collect statistics on multi-local observables' values is used. after a single QCNN layer is applied, several families of feature maps or filters that feed to traditional convolutional network layers or fully connected layers may be applied in a translationally invariant fashion.

Target problems:

i. information extracting, utilizing the noisy quantum circuits hybridization together classical NN.

ii. Learning how retrieving classical information happens from a quantum system's correlations.

Required TFQ functionalities:

a. A classical gradient-based optimizer that is quick.

b. Backpropagation based on quantum expectations.

c. Batch quantum circuit simulator.

d. Hybrid quantum-classical network models.

4.3 IMPLEMENTING PROGRAMMING & FRAMEWORKS

4.3.1 Assemble Circuits In A TensorFlow Graph To Build A QCNN

Quantum data source's certain properties can be detected, like complex simulation from device or quantum sensor by applying the quantum analogue on classical convolutional NN as a simplified Quantum Convolutional Neural Network (QCNN) which is translationally invariant. The cluster state is the quantum data source which might or mightn't has excitation and the task is classifying these states by utilizing SPT phase classification.

First of all, TensorFlow and TensorFlow-quantum will setup to account for version changes for Updating package resources through the following instructions over the command prompt as shown in Figure 4.1:

```
Command Prompt
C:\Users>pip install tensorflow==2.7.0
```

```
Command Prompt
C:\Users>pip install tensorflow-quantum
```

Figure 4.1: Setup TensorFlow and TensorFlow-quantum**.**

Next, the importance TensorFlow as well as module dependencies are being called by the instructions shown in Figure 4.2.

```python
import importlib, pkg_resources
importlib.reload(pkg_resources)

import tensorflow as tf
import tensorflow_quantum as tfq

import cirq
import sympy
import numpy as np

# visualization tools
%matplotlib inline
import matplotlib.pyplot as plt
from cirq.contrib.svg import SVGCircuit
```

Figure 4.2: A piece of program code shows the commands used for importing the importance TensorFlow and the module dependencies.

The preparation of quantum data is the first stage in the QML pipeline process. The aim is to identify a collection of properly and wrongly constructed cluster states on 8 qubits in this quantum dataset, a correctly prepared cluster state is generated by the program code shown in Figure 4.3:

```python
def cluster_state_circuit(bits):
    circuit = cirq.Circuit()
    circuit.append(cirq.H.on_each(bits))
    for this_bit, next_bit in zip(
            bits, bits[1:] + [bits[0]]):
        circuit.append(
            cirq.CZ(this_bit, next_bit))
    return circuit
```

Figure 4.3: A piece of program code shows the commands used for preparing cluster state**.**

The circuits can be assembled in a TensorFlow graph, that is Layer classes with in circuit construction are available in TensorFlow Quantum (TFQ), These layers may append or prepend circuits to input batch, like "tfq.layers.AddCircuitLayer" which originates through "tf.keras.Layer" as shown in Figure 4.4:

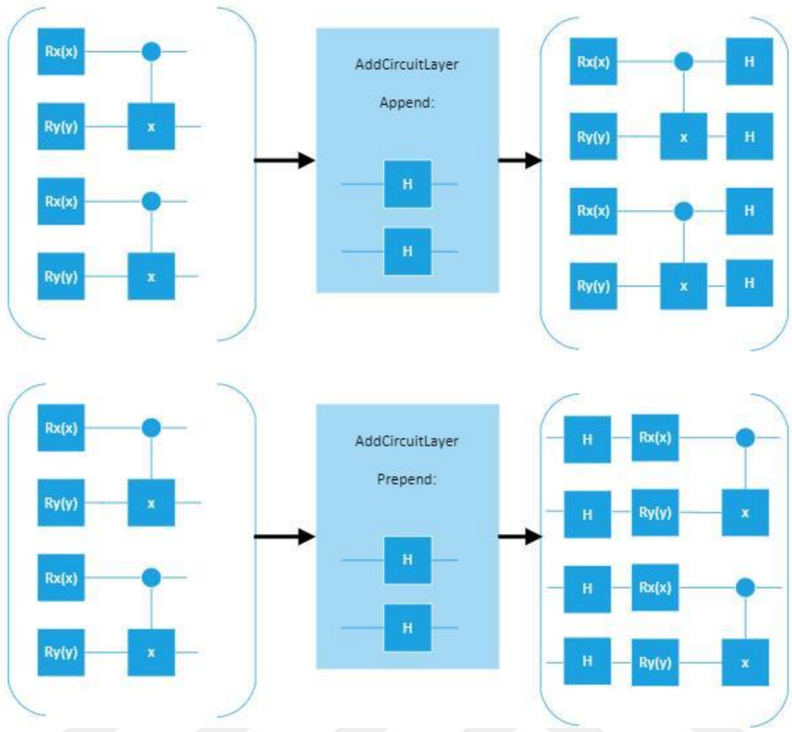


Figure 4.4: The prepending or appending layers to the circuits' input batch**.**

Although the following issue may be performed without "tfq.layers.AddCircuit" is a great way to see how TensorFlow compute graphs may be used to include sophisticated functionality.

4.3.2 Problem Overview

Although the cluster state (which is translationally invariant) is highly entangled, it is not insurmountable for a conventional computer, so it uses a quantum classifier to determine whether or not it is "excited" by preparing and training utilizing a deep MERA-like QCNN architecture to implement classification task aiming reduce entanglement reading out a single qubit. The $R_x(\theta)$ gates, which rotate qubit by a certain amount of $0 \leq \theta \leq 2\pi$ around X-axis inside Bloch sphere [44], will be used to mimic errors in cluster state preparation. If the rotation is greater than a certain threshold, these excitations will be labeled 1; otherwise, they'll be labeled -1. The researcher may conceive of the properly prepared cluster state as the beginning state in the QML pipeline because the status of the appropriately constructed cluster remains the same, constantly followed add the Distinct error states which have different excitation circuits as shown in Figure 4.5:

```
cluster_state_bits = cirq.GridQubit.rect(1, 8)
excitation_input = tf.keras.Input(
    shape =(), dtype=tf.dtypes.string)
cluster_state = tfq.layers.AddCircuit ()(
    excitation_input , prepend=
    cluster_state_circuit(cluster_state_bits))
```

Figure 4.5: A piece of program code shows Excitations will be labeled 1; otherwise, they'll be labeled -1 through rotation a qubit around X-axis inside Bloch sphere [44].

It's worth noting that a standard “Keras” input work as excitation input. To accommodate for our circuit serialization techniques, the data type of the input is a string as shown in Figure 4.6.

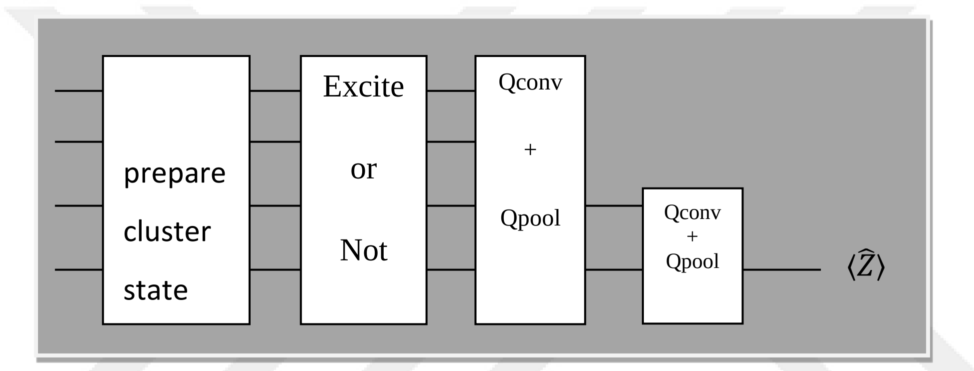


Figure 4.6: Prepare a cluster state.

If the cluster state has gate "cirq.rx" assigned to any of its qubits, then it is classified as "excited".

4.3.3 Building Blocks For TensorFlow

The researcher begins the building of our model after preparing our dataset. All of the models in the current chapter will use quantum pooling and quantum convolution. These operations get visualization over four qubits wherein two qubits unitary $U(\vec{\theta})$ of stride of one is used to create quantum convolution layers as illustrate in Figure 4.7.

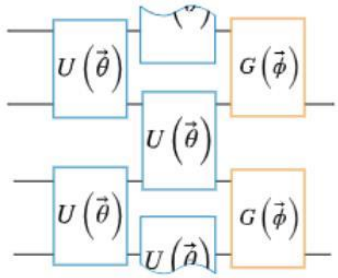


Figure 4.7: A visualization over four qubits of quantum pooling and quantum convolution [44].

Parameters of the unitaries are coupled, similar to traditional convolutional layers, such that every nearest-neighbor qubit's pair undergoes the identical procedure. The quantum convolution and quantum pooling processes are defined in the code shown in Figure 4.8 to disentangle using a new 2 qubits unitary $G(\vec{\phi})$, enabling information to be projected down from two down to one qubit.

```
def quantum_conv_circuit(bits, syms):
    circuit = cirq.Circuit()
    for a, b in zip(bits[0::2], bits[1::2]):
        circuit += two_q_unitary([a, b], syms)
    for a, b in zip(bits[1::2], bits[2::2] + [
    bits[0]]):
        circuit += two_q_unitary([a, b], syms)
    return circuit
def quantum_pool_circuit(srcs, snks, syms):
    circuit = cirq.Circuit()
    for src, snk in zip(srcs, snks):
        circuit += two_q_pool(src, snk, syms)
    return circuit
```

Figure 4.8: A piece of program code shows definition of the quantum pooling and quantum convolution processes.

As clear in program code, while a CNOT represented using "two_q_pool" with target qubits and general one-qubit unitaries on the control, a general parameterized two-qubit unitary constructed by "two_q_unitary", allowing the target basis and the control to be a variational selected.

Before continuing to trace the stages of the QML pipeline, which started with the preparation of quantum data then simulating Errors in cluster state preparation and then the pipeline's initialization stage and the excitation circuits' addition correlating to various levels of error followed by defining each of the parameters of the unitaries And Pooling, ending with measurement and classical post-processing, hybridization's degree of quantum network with the classical network must be determined. This degree of hybridization is summarized in the following three models:

i. Purely quantum CNN.

ii. Hybrid CNN: A shortened QCNN's outputs are fed into a classic densely connected neural network.

iii. Hybrid CNN: A multiple shortened QCNN's outputs are fed into a classic densely connected neural network.

The First Model: uses solely quantum processes to decorrelate the inputs.

Firstly, the cluster state dataset is prepared over N qubits, next the number of qubits in the system is lowered from eight qubits to one qubit utilizing repeatedly the pooling layers and the quantum convolution, thus measuring the expectation of Pauli-Z is applied to this final output qubit to average the quantum model's output. As shown in the program code in Figure 4.9, the function of the "tfq.layers.PQC" object is to enact the Measurement and parameter control.

```
readout_operators = cirq.Z(
    cluster_state_bits [-1])
quantum_model = tfq.layers.PQC(
    create_model_circuit(cluster_state_bits),
    readout_operators)(cluster_state)
qcnn_model = tf.keras.Model(
    inputs =[ excitation_input],
    outputs =[ quantum_model ])
```

Figure 4.9: A piece of program code shows enacting the Measurement and parameter control.

The layers of quantum convolution and quantum pooling are applied sequentially using the "create_model_circuitis" function to be a Purely quantum CNN model output as shown in Figure 4.10.

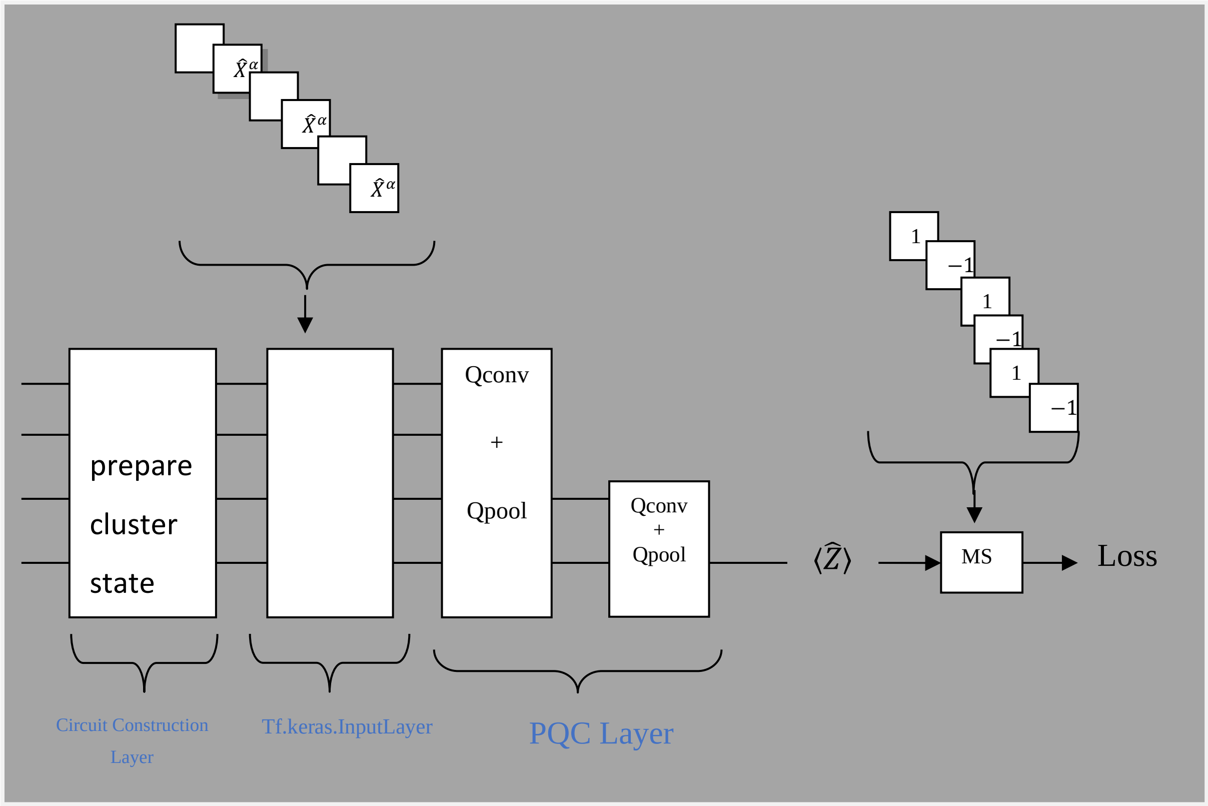


Figure 4.10: Implementing TensorFlow Quantum for Purely quantum CNN model.

Implementing the following in TensorFlow Quantum is one technique to solve this issue:

a. The model takes a circuit tensor as input, which may be either an X gate or an empty circuit on a specific qubit signifying excitation.

b. The remaining quantum components of the model are built using "tfq.layers.AddCircuit" layers.

c. An excited state is denoted by the number 1 and a non- excited state by the number -1 by reading $\langle \hat{Z} \rangle$ wherein "tfq.layers.PQC" layer is utilize from inference.

4.3.4 Data

The data will be created before you start developing the model. Excitations to the cluster state will be the case in this scenario. The "cirq.rx" gate represents excitation, where it is required to consider the rotation an excitation as being large enough and give a sign of 1, otherwise it is considered not excitation and the sign is given -1.

The result looks like classical machine learning wherein to benchmark the model is training, and testing set as shown in Figure 4.11.

```
Input: (0, 0): ──X^0.699── Output: -1
Input: (0, 1): ──X^-0.808── Output: -1
```

Figure 4.11: The Data model benchmark is a training, and testing set.

4.3.5 Define Layers

The layers can be defined in TensorFlow.

4.3.5.1 Cluster State

Quantum circuits programming utilizes the framework provided by Google to use “Cirq” for defining the cluster state depending on "tfq.layers.AddCircuit" functionality which embeds the cluster state into the model. "cirq.GridQubits" Displays a cluster state circuit for a rectangle as shown in Figure 4.12.

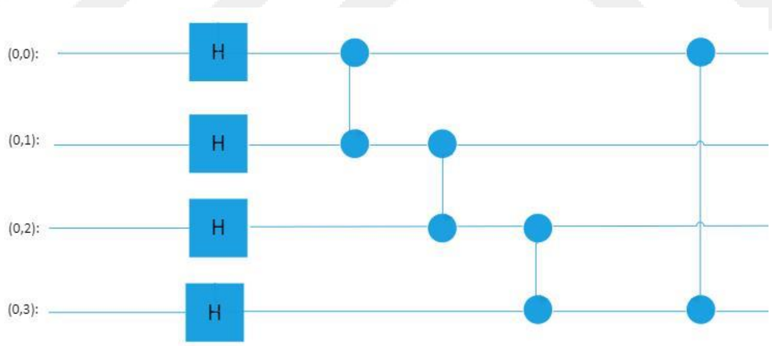


Figure 4.12: A cluster state circuit for a rectangle.

4.3.5.2 QCNN Layers

To construct the model, the author refers to the Cong and Lukin et al. paper [12], in which the layers are defined; to define a one-qubit and two-qubit parameterized unitary matrix, respectively (Figure 4.13 and Figure 4.14); and to implement a pooling operation for a general parameterized two-qubit (Figure 4.15).

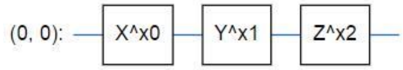


Figure 4.13: The one-qubit unitary circuit.

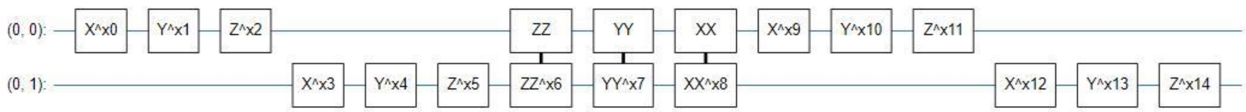


Figure 4.14: The two-qubit unitary circuit.

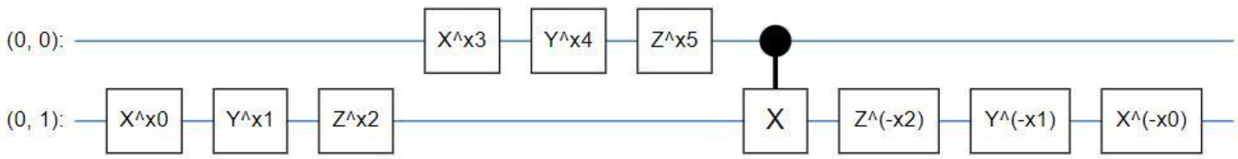


Figure 4.15: The two-qubit pooling circuit.

4.3.5.3 Quantum Convolution

Figure 4.16 illustrates that is by one stride, every pair of adjacent qubits will get two-qubit parameterized unitary applying like defining the 1D quantum convolution in paper of Cong and Lukin et al. [111].

Figure 4.16: The (very horizontal) circuit.

4.3.5.4 Quantum Pooling

Before a pooling component circuit has Examine, the two-qubit pool defined in Figure 4.17 will be utilized pooling from N qubits to N/2 qubits of a quantum pooling layer.

Figure 4.17: Pooling component circuit Examination.

4.3.6 Model Definition

Starting with eight qubits a purely quantum CNN going to be constructed using the defined layers aims to measure $\langle \hat{Z} \rangle$ after it pool down to one as shown in Figure 4.18.

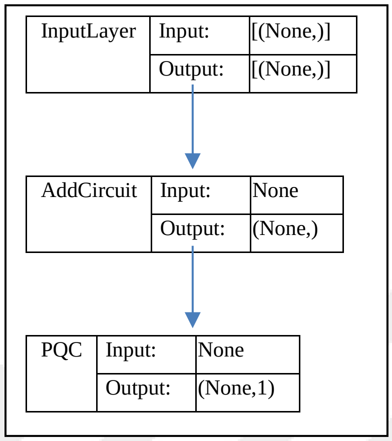
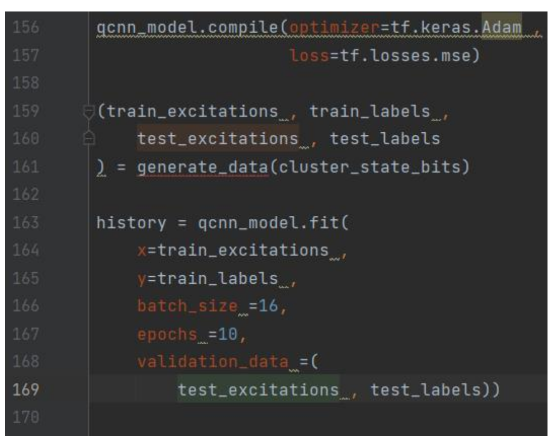


Figure 4.18: The "keras" plot of the model.

4.3.7 Train The Model

Training and validation can be accomplished through the model constructed utilizing standard "Keras" tools as illustrated through program code in Figure 4.19. During training, there is taking the mean squared error between label and model output over each batch from the dataset as a cost function, wherein model output is compared to label for each quantum data point.

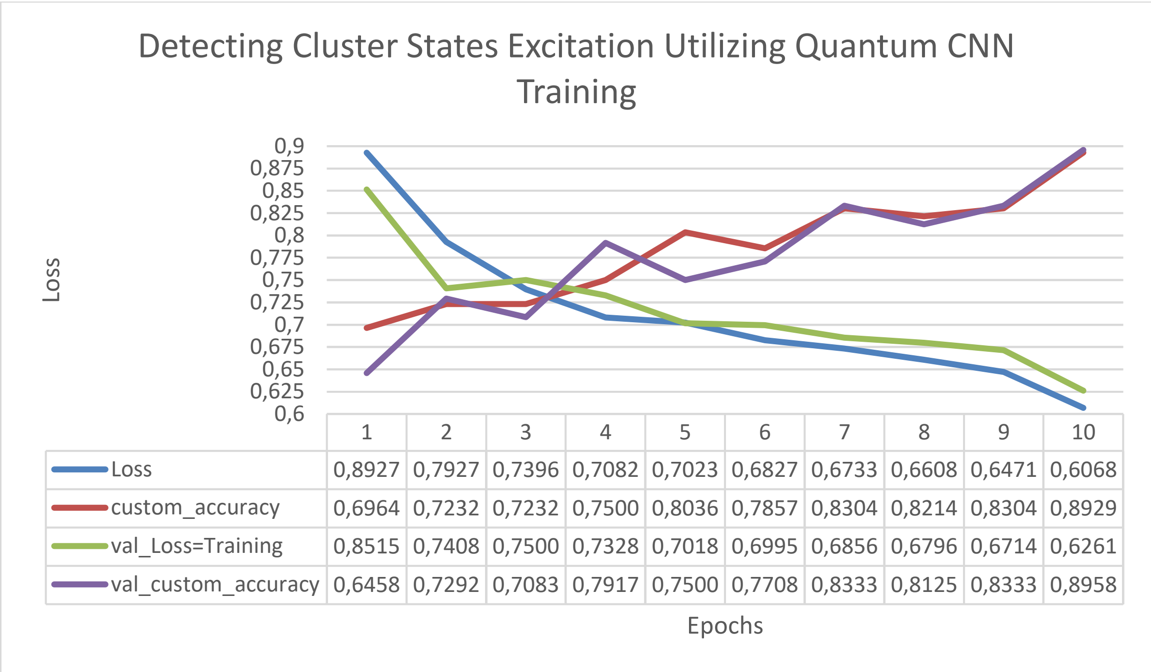


```
qcnn_model.compile(optimizer=tf.keras.Adam ,
                   loss=tf.losses.mse)

(train_excitations , train_labels ,
    test_excitations , test_labels
) = generate_data(cluster_state_bits)

history = qcnn_model.fit(
    x=train_excitations ,
    y=train_labels ,
    batch_size =16,
    epochs =10,
    validation_data =(
        test_excitations , test_labels))
```

Figure 4.19: A piece of program code shows accomplishing the Training and validation.

Building the excitation circuits requires implementation of all the labels associated with the excitation circuits concomitantly side by side on the initial cluster state-input utilizing the "generate_data" function.

Figure 4.20 displays loss plots for validation datasets as well as training in a purely quantum CNN.



	1	2	3	4	5	6	7	8	9	10
Loss	0,8927	0,7927	0,7396	0,7082	0,7023	0,6827	0,6733	0,6608	0,6471	0,6068
custom_accuracy	0,6964	0,7232	0,7232	0,7500	0,8036	0,7857	0,8304	0,8214	0,8304	0,8929
val_Loss=Training	0,8515	0,7408	0,7500	0,7328	0,7018	0,6995	0,6856	0,6796	0,6714	0,6261
val_custom_accuracy	0,6458	0,7292	0,7083	0,7917	0,7500	0,7708	0,8333	0,8125	0,8333	0,8958

Figure 4.20: Loss in a purely quantum CNN.

4.4 HYBRID MODELS

The second model: Is the Hybrid quantum-classical hybrid models HCNN classifier with a single quantum filter (Twist), which will consider being clarified to be: The QCNN may be shortened and the leftover qubits used to measure a vector of operators in place of the quantum layers used to achieve the 1-qubit threshold. As a consequence, the vector of expected values will be input into a traditional densely connected neural network as shown schematically in Figure 4.22. One or two rounds of quantum convolution could have done then fed classical neural network by results instead of going from eight down to one qubit via single quantum convolution.

4.4.1 Hybrid Model With A Single Quantum Filter

This model may be implemented in TFQ by adding a few simple revisions model and using the program code indicated in Figure 4.21.

```
# Build  multi -readout  quantum  layer
readouts = [cirq.Z(bit) for bit in
    cluster_state_bits [4:]]
quantum_model_dual = tfq.layers.PQC(
    multi_readout_model_circuit(
    cluster_state_bits),
    readouts)(cluster_state)
# Build  classical  neural  network  layers
d1_dual = tf.keras.layers.Dense (8)(
    quantum_model_dual)
d2_dual = tf.keras.layers.Dense (1)(d1_dual)
hybrid_model = tf.keras.Model(inputs =[
    excitation_input], outputs =[ d2_dual ])
```

Figure 4.21: A piece of program code shows adding a few simple revisions model by building Classical neural network layers.

"multi_readout_model_circuit" function reduces the system size from eight to four qubits by using just one pooling and convolution cycle in the algorithm. Reading out $\langle \widehat{Z_n} \rangle$ across the board for the lone layer of quantum convolution that had been performed, subsequently feeding a densely connected classical neural network.

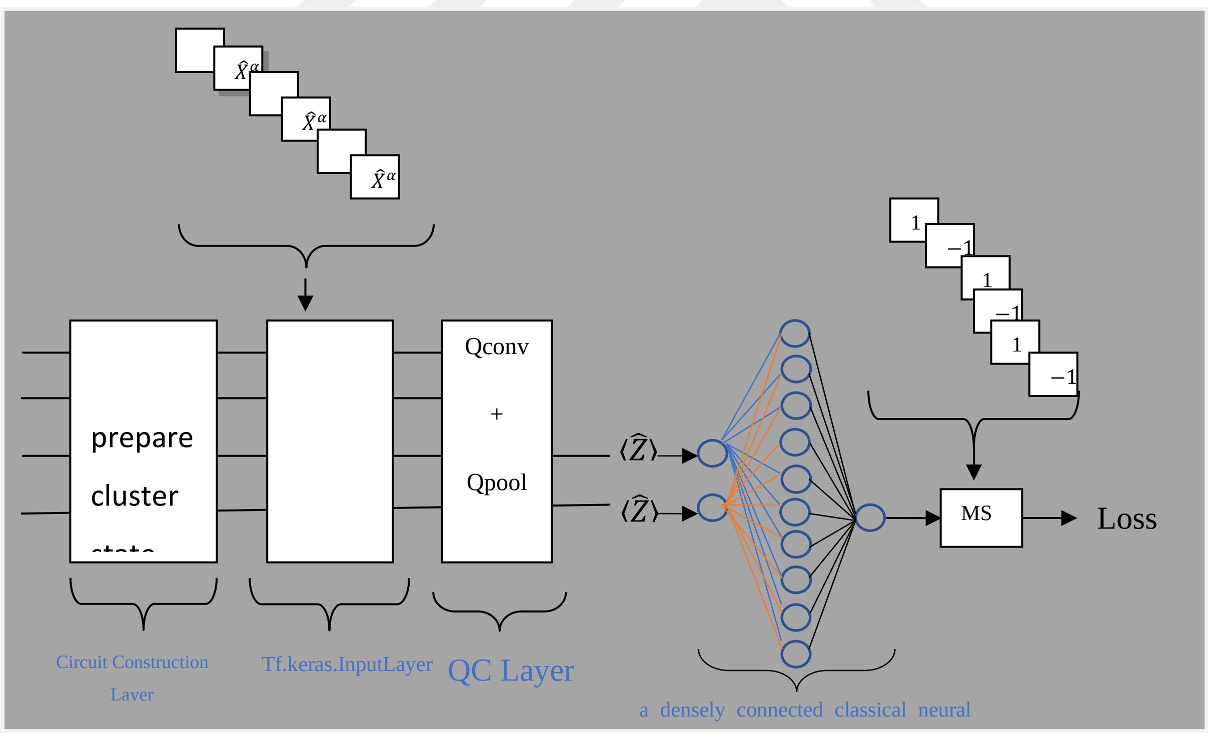


Figure 4.22: Loss measurement for Hybrid model with a single quantum filter.

4.4.1.1 Model Definition

Instead of getting down to 1 qubit to shorten the QCNN, quantum layers may be used to pool information followed by measuring a operators' vector on remaining qubits, as explained in Hybrid quantum-classical hybrid models HCNN classifier using a single quantum filter (Twist). As a consequence, the expectation values' result vector will be supplied to a traditional densely connected neural net, as depicted schematically in

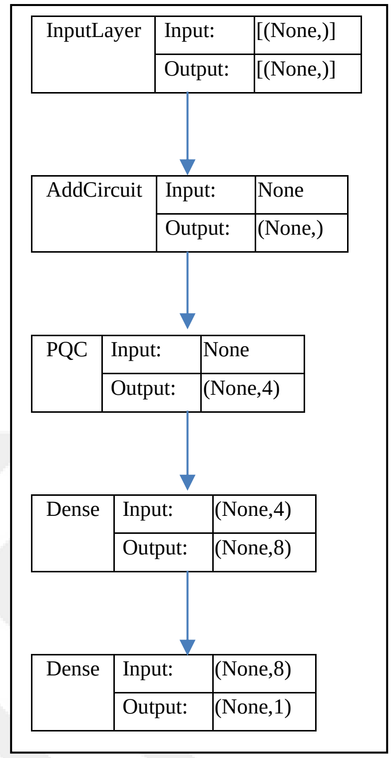


Figure 4.23: Define Hybrid model with a single quantum Twist.

4.4.1.2 Train The Model

The same "Keras" tools may be used to train hybrid model as a purely quantum model, to get Accuracy plots shown in Figure 4.24.

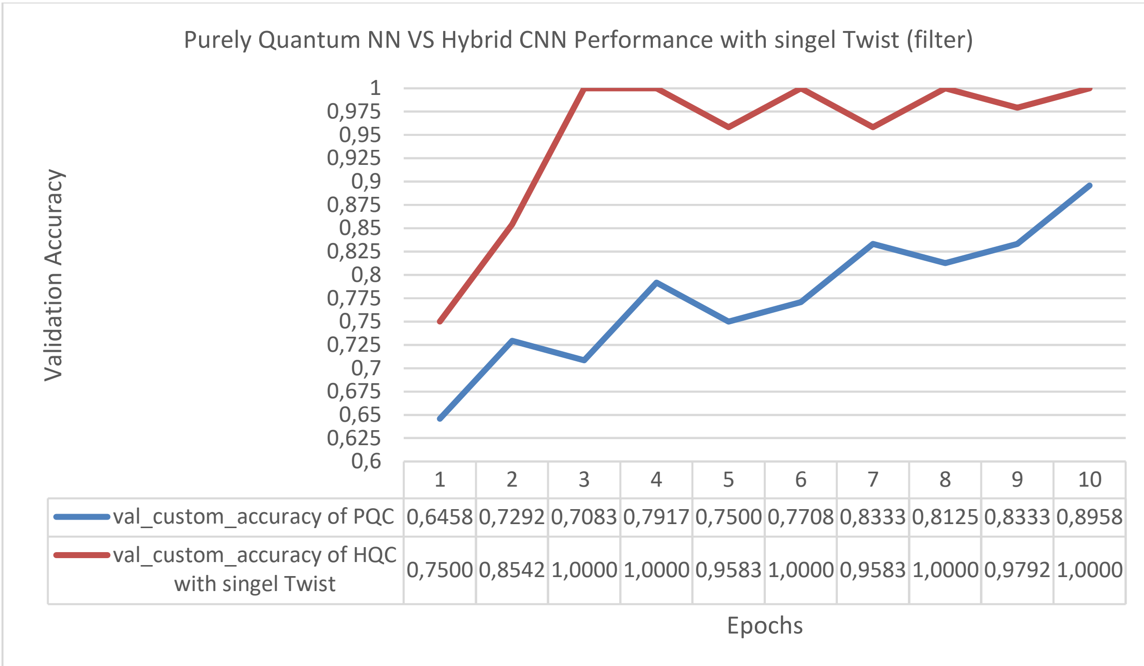


Figure 4.24: Validation Accuracy measurement between purely Quantum vs Hybrid CNN performance.

Note that the purely quantum version converges sluggishly comparing with the hybrid model because of very modest classical assistance.

4.4.2 Hybrid Convolution With Multiple Quantum Filters

The third model: Utilizes a single conventional neural network In order to combine the results of three separate quantum filters as shown in Figure 4.26. The program code implemented in : Achieves this multi-filter architecture.

```
# Build 3 quantum  filters
QCNN_1 = tfq.layers.PQC(
    multi_readout_model_circuit(
    cluster_state_bits),
    readouts)(cluster_state)
QCNN_2 = tfq.layers.PQC(
    multi_readout_model_circuit(
    cluster_state_bits),
    readouts)(cluster_state)
QCNN_3 = tfq.layers.PQC(
    multi_readout_model_circuit(
    cluster_state_bits),
    readouts)(cluster_state)
# Feed  all  QCNNs  into a classical  NN
concat_out = tf.keras.layers.concatenate(
    [QCNN_1 , QCNN_2 , QCNN_3 ])
dense_1 = tf.keras.layers.Dense (8)(concat_out)
dense_2 = tf.keras.layers.Dense (1)(dense_1)
```

Figure 4.25: A piece of program code shows achievement this multi-filter architecture.

This model, which utilizes a mix of a single classical neural network and many quantum convolutions with multi-Twists, is the subject of the emphasis of the topic of this thesis's discussion (filters).

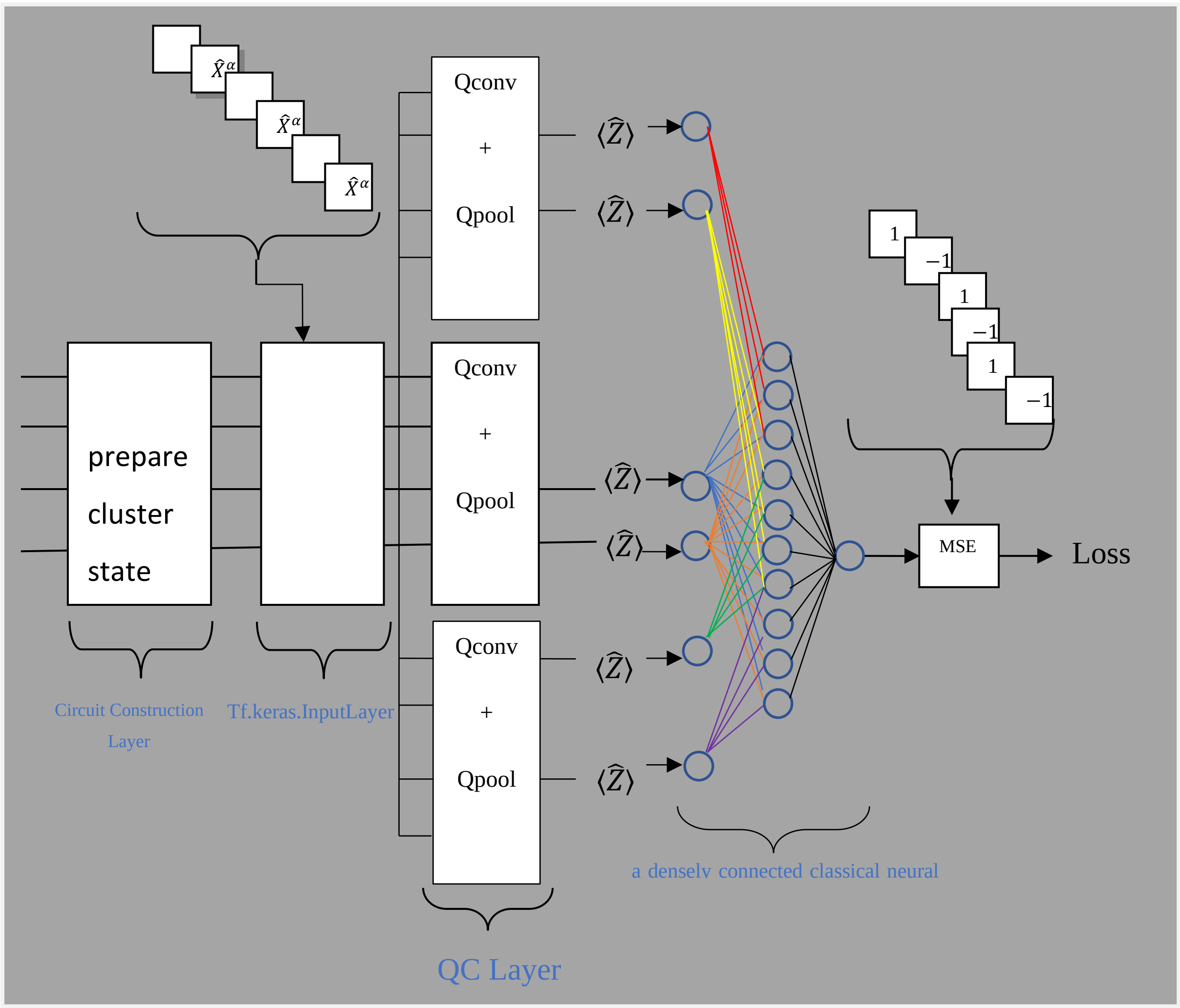


Figure 4.26: An architecture containing numerous quantum convolutions and conventional neural networks, with "Twists" (filters).

4.4.2.1 Model Definition

Utilizes a single conventional neural network In order to combine the results of three separate quantum filters as shown in Figure 4.27.

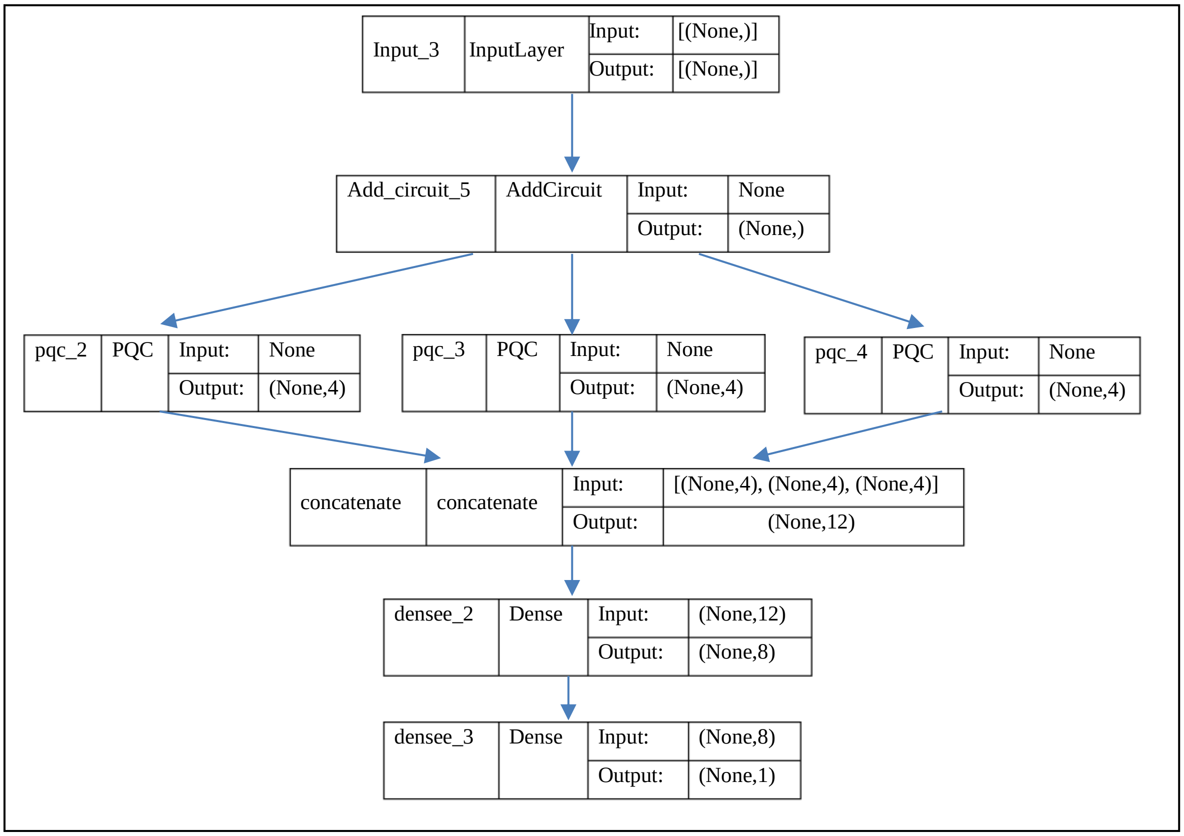


Figure 4.27: The model architecture applies three filters then measure expectation values.

4.4.2.2 Train The Model

The accuracy graphs in Figure 4.28 may be obtained by training the hybrid model using the same "Keras" tools as the purely quantum model. The benefits of experimenting with hybrid quantum-classical architectures for classifying quantum data are illustrated in Figure 4.28, which shows that as accuracy increases, losses in training the model decrease. This is true even when using the same optimization settings as the purely quantum mode.

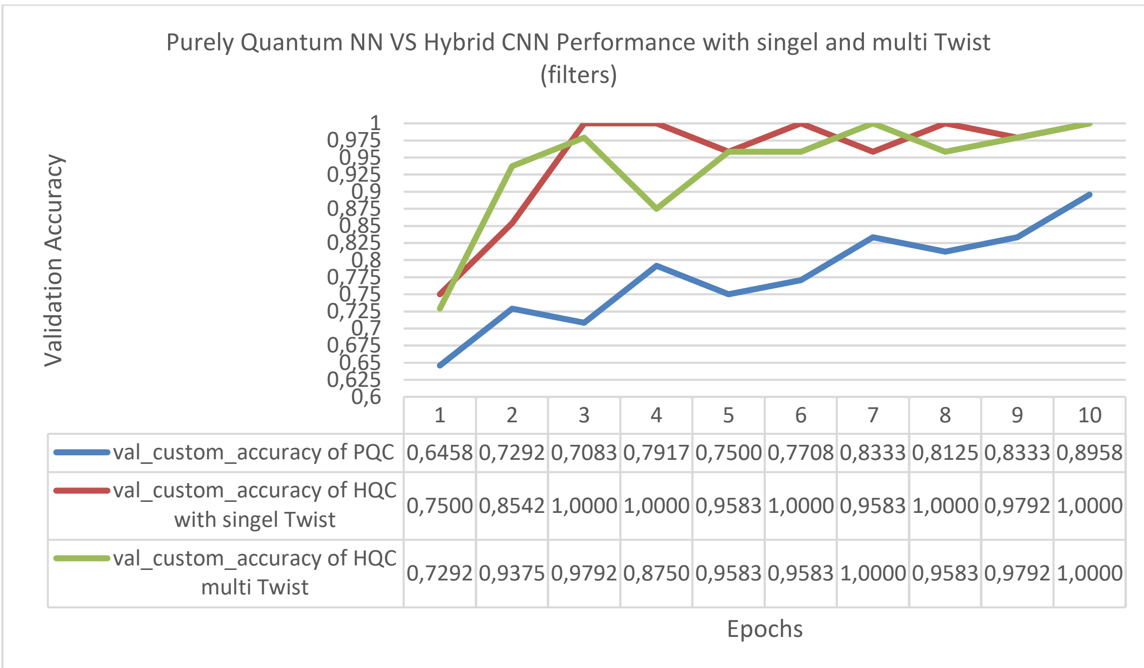


Figure 4.28: Comparison of the Single-Twist and Multi-Twist Validation Accuracy of Purely Quantum and Hybrid CNNs.

5. RESULTS AND DISCUSSION

TensorFlow develops software tools for hybrid quantum-classical models, boosting quantum machine learning (QML) by spreading computations across CPUs, GPUs, and TPUs [112] to incorporate QPUs, automating typical model-building tasks. Before executing quantum components of these models on actual quantum processors, Cirq-TensorFlow with its TensorFlow Quantum TFQ integration is used when dealing with hybrid quantum-classical models in the design, training, and testing stages. The TFQ's principle is based on representing computations as graphs in the form of a pipeline, as illustrated in Figure 5.1, by splitting the model (whether quantum or classical) into layers, the direction of data flow, the flow of computations, and the transmission of instructions according to needs and intermediate outcomes may be manipulated. It's done with the assistance of the tools in TF and Keras, which TFQ has harnessed. Tensors can be used with dataflow graphs by TFQ for the operations (ops) and directed edges of nodes, so it is simple to obtain these graphs' gradients to update the parameters and then distribute these independent graphs nodes to independent devices, including TPUs and GPUs and run them concurrently. It gives a detailed overview of TFQ, focusing on gradient-based methods as important properties, to give a basic level understanding of the framework mechanism, wherein the construction of quantum computing and Quantum circuits work as tensors that ops simulate or execute on a quantum device to convert into classical information. These tensors will be formed after constructing certain circuits using convert "Cirq" objects to TensorFlow string tensors utilizing "tfq.convert_to_ tensor" function. SymPy symbols may be used to parameterize the "cirq.Circuit" objects. State sampling, calculation, expectation value, or simulation are used to transform these tensors into classical information. To solve any issue with QML, all that is necessary is tensor circuits representation and Paulis, as well as operators execution. For in-graph circuit construction, TFQ gives extra ops when dealing with problems during training or inference, just a small component is changed where circuit bulk is static, this has been proven to be useful for ease in handling with TFQ, for instance, one of the functionalities is provided using "tfq.tfq_append_circuitop". TFQ's tools can communicate with both actual quantum hardware and core TensorFlow through Cirq. Importantly, TFQ's APIs let users create a single TensorFlowQuantum application that can then be run locally on a desktop in a PetaFLOP highly parallel and distributed environment. The TFQ's Pipeline, depicted in Figure 5.1, For quantum data, TFQ's end-to-end pipeline of a hybrid quantum-classical discriminative model's training and inference gives a high-level abstract overview of the computational phases required.

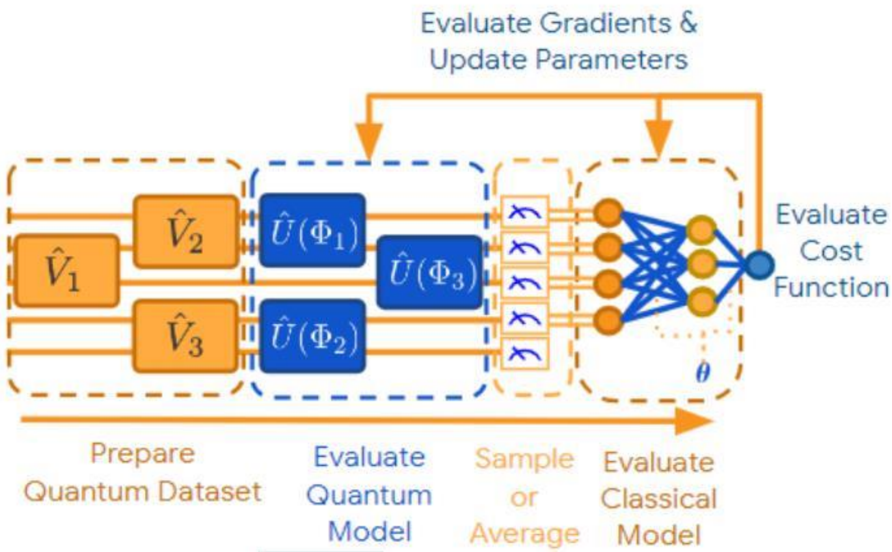


Figure 5.1: Quantum data Flowchart with a hybrid quantum-classical discriminative model's training and inference using TFQ's pipeline [44].

i. Preparing Quantum Dataset:

This might, in general, originate from a black-box source. Due to the fact that contemporary quantum computers are unable to ingest quantum data based on outside sources, quantum circuits should describe that create data. Unparameterized "cirq.Circuit" objects are used to create quantum datasets, where "tfq.convert_to_tensor" is used to inject data into the graph.

ii. Quantum Model Evaluation:

Depending on the knowledge of the structure of quantum data, parameterized quantum models may be chosen from a variety of categories. The model's goal is getting quantum computation to extract data from a hidden quantum subsystem or subspace. This data has hidden label parameters in discriminative learning. The quantum model disentangles a data input extracting quantum non-local subsystem, post-processing quantum models are created (because of leaving concealed information in classical correlations available to local measurements) using one of "cirq.Circuit" objects that are coupled to quantum data sources with "tfq.AddCircuitlayer" objects which are "SymPy" symbols.

iii. Average or Sample:

Quantum state measurements retrieve classical information from classical random variables in samples form. The quantum state and the observed observable both influence the distribution of the values from this random variable.

iv. Classical Model Evaluation:

After retrieving classical data in the form may be used for more classical post-processing. Deep neural networks utilize for distilling Classical correlations because the recovered data might still be kept through these correlations among expectations measured. Quantum models directly can be applied to "tf.keras.layers.Layer" objects like "tf.keras.layers.Dense" in the same way completely compatible with core TensorFlow.

v. Evaluate Cost Function:

A cost function is constructed based on the outcomes of classical post-processing. If there was a label on the quantum data, this may be based on classification accuracy or other criteria. all losses in "tf.keras.losses" module can be accessed by tracking TFQ's Pipeline the model produced in previous phases within "atf.keras.Model".

vi. Evaluate Gradients & Update Parameters:

Following the evaluation of cost function, parameters in TFQ's pipeline are modified with way that reduces the cost. Gradient descent is the most popular method for doing this. The "tfq.differentiators" differentiator interface provides quantum operations' derivatives to TensorFlow backpropagation engine to allow gradient descent. By using hybrid quantum-classical backpropagation, quantum data may be used to modify parameters of both classical and quantum models.

The researcher made several attempts to train the models during different Epochs, wherein based on the supervised learning method to learn a quantum model for processing quantum data in three approaches: a purely quantum model, a hybrid quantum model with a single filter (Twist) and a hybrid quantum model with a multi-filter (multi-Twist). The researcher has noticed that the hybrid model gives an accurate result of 100% accuracy during the Epochs period of no more than 7

Epochs in the case of training a multi-filter model (multi-Twist) and the hybrid model is not decorated with more than 3 Epochs during the case of training a single-twist model compared to its purely quantum model, which gives an accuracy result of no more than 89.58%. Since the amount of loss in the model training is inversely proportional to the amount of accuracy of the model training results. And thus, the actual benefit has been achieved from merging the classical model with the quantum model in the hybrid quantum-classical discriminative model for quantum data using an algorithm that reduces the amount of loss in training the model by reducing entanglement in quantum data and provides accuracy and speed in training this model accessing to results during (epochs) is reasonable.

6. CONCLUSION AND RECOMMENDATIONS

The rapid advancement of quantum devices in recent years has promoted quantum applications' development. In October 2017 and aiming to accelerate the quantum simulation algorithms' development for materials research and chemistry, OpenFermion was the first software library published by Google AI Quantum. TensorFlow Quantum, on the other hand, has been designed to speed up quantum machine learning algorithms' development for variety of applications. The researcher has implemented the preprocessing on QML models by performing capacity coding during the application and testing of the state preparation method on the TensorFlow Quantum framework in order to reach trained models with several steps which make it more powerful by analyzing the methods used in training the quantum models to facilitate understanding of preparing quantum model processing methods in the future, *Future Work* in this field will need the development of multiclass classification algorithms that would be sufficiently developed to be used to works such as this one.

The researcher anticipates that the framework will continue to evolve in response to the requirements of the research community and the availability of additional quantum technology, since quantum machine learning is a relatively new and intriguing field of study. The researcher takes a look around the software's theory. Quantum neural networks are defined simply the combinations of unitary matrices with parameters. Loss function is expressed like the inner product, which defines samples and expected values. Gradients may be defined once expectation values and quantum neural networks have been determined. Finally, one of the main components of TFQ, hybrid quantum-classical backpropagation, which is formalized by combining quantum and classical neural networks. The researcher has covered TensorFlow, a flexible data flow-based programming approach, as well as single-machine and distributed TensorFlow implementations. The approach is based on Google's real-world experience conducting research and delivering over one hundred machine learning initiatives across a broad spectrum of goods and services. The researcher made several attempts to train the models at various Epochs and he observed that the hybrid model achieves 100% accuracy during an Epoch period of no more than seven in the case of training a multi-filter model (multi-Twist) and no more than three in the case of training a single-twist model, in comparison to its purely quantum model, which achieved an accuracy of no more than 89.58%. As a result, the researcher proposes to stop training the model after it achieves 100 percent accuracy in order to save time of the training algorithm's implementing; thus, the true benefit of combining the classical and quantum models in a hybrid model has been realized through the use of an algorithm that provides accuracy and speed in training this model, as well as reasonable access to results during (epochs). The goal is that comparable outcomes or patterns emerge for the learning outcomes. The researcher may state categorically that the approaches used here are not guaranteed to perform identically, but with foresight, considerable similarities are expected.

REFERENCES

[1] I. Bengtsson and K. Zyczkowski, *Geometry of quantum states: An introduction to quantum entanglement*. 2017. doi: 10.1017/9781139207010.

[2] D. P. Kingma and J. Lei Ba, “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION”.

[3] M. A. Nielsen *et al.*, “Quantum dynamics as a physical resource,” *Phys Rev A*, vol. 67, no. 5, 2003, doi: 10.1103/physreva.67.052301.

[4] S. Kullback and R. A. Leibler, “On Information and Sufficiency,” *The Annals of Mathematical Statistics*, vol. 22, no. 1, 1951, doi: 10.1214/aoms/1177729694.

[5] M. Peskin and D. Schroeder, *An Introduction To Quantum Field Theory (Frontiers in Physics)*. 2015.

[6] C. Cherubini and S. Filippi, “Lagrangian field theory of reaction-diffusion,” *Phys Rev E Stat Nonlin Soft Matter Phys*, vol. 80, no. 4, 2009, doi: 10.1103/PhysRevE.80.046117.

[7] D. Melikhov, “Analytic Properties of Triangle Feynman Diagrams in Quantum Field Theory,” *Particles*, vol. 3, no. 1, 2020, doi: 10.3390/particles3010009.

[8] A. Pais, “Theoretical particle physics,” *Rev Mod Phys*, vol. 71, no. SUPPL. 2, 1999, doi: 10.1103/revmodphys.71.s16.

[9] J. A. R. Shea and E. Neuscamman, “Communication: A mean field platform for excited state quantum chemistry,” *Journal of Chemical Physics*, vol. 149, no. 8, 2018, doi: 10.1063/1.5045056.

[10] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, “Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms,” *Adv Quantum Technol*, vol. 2, no. 12, 2019, doi: 10.1002/qute.201900070.

[11] Y. S. Weinstein, W. G. Brown, and L. Viola, “Parameters of pseudorandom quantum circuits,” *Phys Rev A*, vol. 78, no. 5, 2008, doi: 10.1103/PhysRevA.78.052332.

[12] I. Cong, S. Choi, and M. D. Lukin, “Quantum convolutional neural networks,” *Nat Phys*, vol. 15, no. 12, 2019, doi: 10.1038/s41567-019-0648-8.

[13] V. Giovannetti, S. Lloyd, and L. Maccone, “Quantum-enhanced measurements: Beating the standard quantum limit,” *Science*, vol. 306, no. 5700. 2004. doi: 10.1126/science.1104149.

[14] N. Moll *et al.*, “Quantum optimization using variational algorithms on near-term quantum devices,” *Quantum Science and Technology*, vol. 3, no. 3. 2018. doi: 10.1088/2058-9565/aab822.

[15] Y. Levine, O. Sharir, N. Cohen, and A. Shashua, “Quantum Entanglement in Deep Learning Architectures,” *Phys Rev Lett*, vol. 122, no. 6, 2019, doi: 10.1103/PhysRevLett.122.065301.

[16] J. B. Altepeter, E. R. Jeffrey, P. G. Kwiat, S. Tanzilli, N. Gisin, and A. Acín, “Experimental methods for detecting entanglement,” *Phys Rev Lett*, vol. 95, no. 3, 2005, doi: 10.1103/PhysRevLett.95.033601.

[17] J. R. Johansson, P. D. Nation, and F. Nori, “QuTiP: An open-source Python framework for the dynamics of open quantum systems,” *Comput Phys Commun*, vol. 183, no. 8, 2012, doi: 10.1016/j.cpc.2012.02.021.

[18] D. Sierra-Sosa, M. Telahun, and A. Elmaghraby, “TensorFlow Quantum: Impacts of Quantum State Preparation on Quantum Machine Learning Performance,” *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3040798.

[19] C. Y. Huang, X. Chen, and F. L. Lin, “Symmetry-protected quantum state renormalization,” *Phys Rev B Condens Matter Mater Phys*, vol. 88, no. 20, 2013, doi: 10.1103/PhysRevB.88.205124.

[20] J. Preskill, “Quantum Computing in the NISQ era and beyond,” 2018. doi: 10.22331/q-2018-08-06-79.

[21] E. Farhi, J. Goldstone, and S. Gutmann, “A Quantum Approximate Optimization Algorithm,” 2014.

[22] J. Romero, J. P. Olson, and A. Aspuru-Guzik, “Quantum autoencoders for efficient compression of quantum data,” *Quantum Sci Technol*, vol. 2, no. 4, 2017, doi: 10.1088/2058-9565/aa8072.

[23] E. Farhi and H. Neven, “Classification with Quantum Neural Networks on Near Term Processors,” 2018.

[24] P. D. Johnson, J. Romero, J. Olson, Y. Cao, and A. Aspuru-Guzik, “QVECTOR: an algorithm for device-tailored quantum error correction,” 2017.

[25] R. P. Feynman, “Space-time approach to non-relativistic quantum mechanics,” *Rev Mod Phys*, vol. 20, no. 2, 1948, doi: 10.1103/RevModPhys.20.367.

[26] P. Benioff, “The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines,” *J Stat Phys*, vol. 22, no. 5, 1980, doi: 10.1007/BF01011339.

[27] P. M. Lavrov, “Covariant quantization of Yang-Mills theory in the first order formalism,” *Physics Letters, Section B: Nuclear, Elementary Particle and High-Energy Physics*, vol. 816, 2021, doi: 10.1016/j.physletb.2021.136182.

[28] T. Bollinger, “Yuri Manin on Biomolecules as Quantum Computers,” *Terry’s Archive Online*, vol. 2021, no. 02, 2021, doi: 10.48034/20210205.

[29] M. Zidan *et al.*, “A quantum algorithm based on entanglement measure for classifying Boolean multivariate function into novel hidden classes,” *Results Phys*, vol. 15, 2019, doi: 10.1016/j.rinp.2019.102549.

[30] G. Burkard and D. Loss, “Spin qubits in solid-state structures,” *Europhysics News*, vol. 33, no. 5, 2002, doi: 10.1051/epn:2002503.

[31] N. Kerker, R. Röpke, L. M. Steinert, A. Pooch, and A. Stibor, “Quantum decoherence by Coulomb interaction,” *New J Phys*, vol. 22, no. 6, 2020, doi: 10.1088/1367-2630/ab8efc.

[32] S. Kanno, J. Soda, and J. Tokuda, “Noise and decoherence induced by gravitons,” *Physical Review D*, vol. 103, no. 4, 2021, doi: 10.1103/PhysRevD.103.044017.

[33] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Phys Rev Lett*, vol. 113, no. 3, 2014, doi: 10.1103/PhysRevLett.113.130503.

[34] P. Rebentrost, M. Schuld, L. Wossnig, F. Petruccione, and S. Lloyd, "Quantum gradient descent and Newton's method for constrained polynomial optimization," *New J Phys*, vol. 21, no. 7, 2019, doi: 10.1088/1367-2630/ab2a9e.

[35] V. Havlíček *et al.*, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, 2019, doi: 10.1038/s41586-019-0980-2.

[36] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, "Quantum embeddings for machine learning," Jan. 2020.

[37] S. Yen-Chi Chen, C.-M. Huang, C.-W. Hsing, and Y.-J. Kao, "Hybrid quantum-classical classifier based on tensor network and variational quantum circuit."

[38] N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada, and S. Lloyd, "Continuous-variable quantum neural networks," *Phys Rev Res*, vol. 1, no. 3, 2019, doi: 10.1103/PhysRevResearch.1.033063.

[39] M. Telahun, "Exploring Information for Quantum Machine Learning Models," 2020, doi: https://doi.org/10.18297/etd/3433.

[40] V. Nair and G. E. Hinton, "Rectified linear units improve Restricted Boltzmann machines," 2010.

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, 2014.

[42] M. A. Nielsen, I. Chuang, and L. K. Grover, " Quantum Computation and Quantum Information ," *Am J Phys*, vol. 70, no. 5, 2002, doi: 10.1119/1.1463744.

[43] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers*. 2019.

[44] M. Broughton *et al.*, “TensorFlow Quantum: A Software Framework for Quantum Machine Learning.”

[45] M. A. Nielsen, “Cluster-state quantum computation,” *Reports on Mathematical Physics*, vol. 57, no. 1, 2006, doi: 10.1016/S0034-4877(06)80014-5.

[46] R. Raussendorf and H. J. Briegel, “A one-way quantum computer,” *Phys Rev Lett*, vol. 86, no. 22, 2001, doi: 10.1103/PhysRevLett.86.5188.

[47] R. Raussendorf, D. E. Browne, and H. J. Briegel, “The one-way quantum computer-a non-network model of quantum computation,” 2001.

[48] M. A. Nielsen, “Optical quantum computation using cluster states,” 2004.

[49] D. E. Browne and T. Rudolph, “Resource-efficient linear optical quantum computation,” 2005.

[50] P. Walther *et al.*, “Experimental one-way quantum computing,” *Nature*, vol. 434, no. 7030, 2005, doi: 10.1038/nature03347.

[51] M. A. Nielsen, “Universal quantum computation using only projective measurement, quantum memory, and preparation of the |0 state,” 2001.

[52] D. W. Leung, “Two-qubit Projective Measurements are Universal for Quantum Computation.”

[53] D. W. Leung, “Quantum computation by measurements,” 2004.

[54] J.-L. Brylinski and R. Brylinski, “UNIVERSAL QUANTUM GATES,” 2001.

[55] E. Knill, “Approximation by Quantum Circuits,” 1995.

[56] M. A. Nielsen and C. M. Dawson, “Fault-tolerant quantum computation with cluster states,” 2004.

[57] H. J. Briegel and R. Raussendorf, “Persistent entanglement in arrays of interacting particles,” *Phys Rev Lett*, vol. 86, no. 5, 2001, doi: 10.1103/PhysRevLett.86.910.

[58] X. Zhou, D. W. Leung, and I. L. Chuang, “Methodology for quantum logic gate construction,” 2000.

[59] A. Einstein, B. Podolsky, and N. Rosen, “Can quantum-mechanical description of physical reality be considered complete?,” *Physical Review*, vol. 47, no. 10, 1935, doi: 10.1103/PhysRev.47.777.

[60] G. Vidal, “Efficient classical simulation of slightly entangled quantum computations,” 2003.

[61] F. Verstraete, M. M. Wolf, D. Perez-Garcia, and J. I. Cirac, “Criticality, the area law, and the computational power of PEPS,” 2006.

[62] Y.-Y. Shi, L.-M. Duan, and G. Vidal, “Classical simulation of quantum many-body systems with a tree tensor network,” 2006.

[63] T. J. Osborne, “The ground state of a class of noncritical 1D quantum spin systems can be approximated efficiently,” 2007.

[64] U. Schollwöck, “The density-matrix renormalization group *,” 2004.

[65] G. Vidal, “Class of quantum Many-Body states that can be efficiently simulated,” *Phys Rev Lett*, vol. 101, no. 11, 2008, doi: 10.1103/PhysRevLett.101.110501.

[66] S. Stlund and S. Rommer, “Thermodynamic limit of density matrix renormalization,” *Phys Rev Lett*, vol. 75, no. 19, pp. 3537–3540, 1995, doi: 10.1103/PhysRevLett.75.3537.

[67] R. N. C. Pfeifer, G. Evenbly, and G. Vidal, “Entanglement renormalization, scale invariance, and quantum criticality,” *Phys Rev A*, vol. 79, no. 4, 2009, doi: 10.1103/PhysRevA.79.040301.

[68] G. Vidal, “Entanglement renormalization,” 2006.

[69] G. Vidal, J. I. Latorre, E. Rico, and A. Kitaev, “Entanglement in quantum critical phenomena,” 2002.

[70] P. Calabrese and J. Cardy, “Entanglement Entropy and Quantum Field Theory,” 2008.

[71] A. R. Its, B.-Q. Jin, V. E. Korepin, and C. N. Yang, “Entanglement in XY Spin Chain,” 2004.

[72] N. Maskara, A. Kubica, and T. Jochym-O’connor, “Advantages of versatile neural-network decoding for topological codes.”

[73] P. W. Shor, “Scheme for reducing decoherence in quantum computer memory,” *Phys Rev A (Coll Park)*, vol. 52, no. 4, 1995, doi: 10.1103/PhysRevA.52.R2493.

[74] D. Gottesman, “Class of quantum error-correcting codes saturating the quantum Hamming bound,” *Phys Rev A (Coll Park)*, vol. 54, no. 3, pp. 1862–1868, Sep. 1996, doi: 10.1103/PhysRevA.54.1862.

[75] P. Iyer and D. Poulin, “Hardness of Decoding Quantum Stabilizer Codes,” *IEEE Trans Inf Theory*, vol. 61, no. 9, pp. 5209–5223, 2015, doi: 10.1109/TIT.2015.2422294.

[76] A. Y. Kitaev, “Fault-tolerant quantum computation by anyons,” *Ann Phys (N Y)*, vol. 303, no. 1, pp. 2–30, 2003, doi: 10.1016/S0003-4916(02)00018-0.

[77] S. B. Bravyi, A. Y. Kitaev, and L. D. Landau, “Quantum codes on a lattice with boundary. *,” 1998.

[78] H. Bombin and M. A. Martin-Delgado, “Topological Quantum Distillation,” 2007.

[79] H. Bombín, D. A. Lidar, and T. A. Brun, “An Introduction to Topological Quantum Codes.”

[80] J. Haah, “Local stabilizer codes in three dimensions without string logical operators,” *Phys Rev A (Coll Park)*, vol. 83, no. 4, p. 42330, Apr. 2011, doi: 10.1103/PhysRevA.83.042330.

[81] S. Bravyi, M. Suchara, and A. Vargo, “Efficient algorithms for maximum likelihood decoding in the surface code,” *Phys Rev A (Coll Park)*, vol. 90, no. 3, p. 32326, Sep. 2014, doi: 10.1103/PhysRevA.90.032326.

[82] A. S. Darmawan and D. Poulin, “Linear-time general decoding algorithm for the surface code.”

[83] C. G. Brell, S. T. Flammia, S. D. Bartlett, and A. C. Doherty, "Toric codes and quantum doubles from two-body Hamiltonians," *New J Phys*, vol. 13, 2011, doi: 10.1088/1367-2630/13/5/053039.

[84] N. Delfosse, "Decoding color codes by projection onto surface codes," *Phys Rev A (Coll Park)*, vol. 89, no. 1, p. 12317, Jan. 2014, doi: 10.1103/PhysRevA.89.012317.

[85] N. H. Nickerson and B. J. Brown, "Analysing correlated noise on the surface code using adaptive decoding algorithms," 2019. doi: 10.22331/q-2019-04-08-131.

[86] J. Harrington, "Analysis of quantum error-correcting codes: symplectic lattice codes and toric codes," *Citeseer*, vol. 2004, 2004.

[87] M. Herold, E. T. Campbell, J. Eisert, and M. J. Kastoryano, "Cellular-automaton decoders for topological quantum memories," vol. 1, p. 15010, 2015, doi: 10.1038/npjqi.2015.10.

[88] K. Duivenvoorden, N. P. Breuckmann, and B. M. Terhal, "Renormalization group decoder for a four-dimensional toric code."

[89] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, "Quantum Error Correction and Orthogonal Geometry," *Phys Rev Lett*, vol. 78, no. 3, pp. 405–408, Jan. 1997, doi: 10.1103/PhysRevLett.78.405.

[90] T. J. Yoder and I. H. Kim, "The surface code with a twist," *Quantum*, vol. 1, pp. 6–8, 2017, doi: 10.22331/q-2017-04-25-2.

[91] F. Wilczek, "Quantum Mechanics of Fractional-Spin Particles," *Phys Rev Lett*, vol. 49, no. 14, pp. 957–959, Oct. 1982, doi: 10.1103/PhysRevLett.49.957.

[92] J. Preskill, "Lecture Notes for Ph219/CS219: Topological quantum computation. Chapter 9," *Lecture Notes for Ph219/CS219*, vol. 9, no. June, p. 68, 2004.

[93] H. Bombin and M. A. Martin-Delgado, "Exact topological quantum order in $D=3$ and beyond: Branyons and brane-net condensates," *Phys Rev B*, vol. 75, no. 7, p. 75103, Feb. 2007, doi: 10.1103/PhysRevB.75.075103.

[94] M. Levin, "Protected Edge Modes without Symmetry", doi: 10.1103/PhysRevX.3.021009.

[95] H. Bombin, “Topological Order with a Twist: Ising Anyons from an Abelian Model,” *Phys Rev Lett*, vol. 105, no. 3, p. 30403, Jul. 2010, doi: 10.1103/PhysRevLett.105.030403.

[96] H. Bombín, “Structure of 2D Topological Stabilizer Codes,” *Commun Math Phys*, vol. 327, no. 2, pp. 387–432, 2014, doi: 10.1007/s00220-014-1893-4.

[97] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, “Quantum Machine Learning,” 2018.

[98] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.

[99] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2012, vol. 2, pp. 1097–1105.

[100] R. P. Feynman, “Simulating physics with computers,” *International Journal of Theoretical Physics*, vol. 21, no. 6, pp. 467–488, 1982, doi: 10.1007/BF02650179.

[101] V. Giovannetti, S. Lloyd, and L. Maccone, “Quantum Random Access Memory,” *Phys Rev Lett*, vol. 100, no. 16, p. 160501, Apr. 2008, doi: 10.1103/PhysRevLett.100.160501.

[102] H. Chen, · L Wossnig, · S Severini, · H Neven, and · M Mohseni, “Universal discriminative quantum neural networks”, doi: 10.1007/s42484-020-00025-7.

[103] G. Verdon *et al.*, “Learning to learn with quantum neural networks via classical neural networks.”

[104] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, “Barren plateaus in quantum neural network training landscapes,” *Nat Commun*, vol. 9, no. 1, 2018, doi: 10.1038/s41467-018-07090-4.

[105] J. Schulman, N. Heess, T. Weber, and P. Abbeel, “Gradient Estimation Using Stochastic Computation Graphs.”

[106] A. Meurer *et al.*, “SymPy: Symbolic computing in python,” *PeerJ Comput Sci*, vol. 2017, no. 1, 2017, doi: 10.7717/peerj-cs.103.

[107] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Comput Sci Eng*, vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55.

[108] V. Bergholm *et al.*, "PennyLane: Automatic differentiation of hybrid quantum-classical computations."

[109] E. Grant *et al.*, "ARTICLE Hierarchical quantum classifiers", doi: 10.1038/s41534-018-0116-9.

[110] R. Schwartz-Ziv and N. Tishby, "OPENING THE BLACK BOX OF DEEP NEURAL NETWORKS VIA INFORMATION," 2017.

[111] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nat Phys*, vol. 15, no. 12, 2019, doi: 10.1038/s41567-019-0648-8.

[112] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems."